

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	21907753	(@ad <"20012717" @rlad <"20012717")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/04/23 22:00
S2	2	((("6498861") or ("6070159")).PN.	USPAT	OR	OFF	2005/04/23 22:00
S3	1010	(file NEAR3 conversion).ab. AND (@ad < "20011503" @rlad < "20011503")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/04/23 22:31
S4	258	(file NEAR3 conversion).ab. AND (@ad < "20011503" @rlad < "20011503") AND (display\$4)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/04/23 22:32
S5	23	(file NEAR3 conversion).ab. AND (@ad < "20011503" @rlad < "20011503") AND (display\$4)	US-PGPUB	OR	OFF	2005/04/23 22:32
S6	31	(file NEAR3 conversion).ab. AND (@ad < "20011503" @rlad < "20011503") AND (display\$4)	USPAT	OR	OFF	2005/04/23 23:04
S7	0	(file NEAR3 conversion).ab. AND (@ad < "20011503" @rlad < "20011503") AND (display\$4) AND ((cell ADJ phone) cellphone cell-phone)	USPAT	OR	OFF	2005/04/23 23:06
S8	1	(file NEAR3 conversion).ab. AND (@ad < "20011503" @rlad < "20011503") AND (display\$4) AND ((cell ADJ phone) cellphone cell-phone PDA)	USPAT	OR	OFF	2005/04/23 23:08
S9	35	(file NEAR3 conversion) AND (@ad < "20011503" @rlad < "20011503") AND (display\$4) AND ((cell ADJ phone) cellphone cell-phone PDA)	USPAT	OR	OFF	2005/04/23 23:15
S10	0	(image ADJ size) AND (chracter\$3 WITH count\$3 WITH line\$1) AND (@ad < "20011503" @rlad < "20011503") AND (display\$4) AND ((cell ADJ phone) cellphone cell-phone PDA)	USPAT	OR	ON	2005/04/23 23:17
S11	0	(image ADJ size) AND (character\$3 WITH count\$3 WITH line\$1) AND (@ad < "20011503" @rlad < "20011503") AND (display\$4) AND ((cell ADJ phone) cellphone cell-phone PDA)	USPAT	OR	ON	2005/04/23 23:17

S12	5	(image ADJ size) AND (character\$3 WITH count\$3 WITH line\$1) AND (@ad < "20011503" @rlad < "20011503") AND (display\$4) AND ((cell ADJ phone) cellphone cell-phone PDA wireless)	USPAT	OR	ON	2005/04/23 23:21
S13	5	(image ADJ size) AND (character\$3 WITH count\$3 WITH line\$1) AND (@ad < "20011503" @rlad < "20011503") AND ((cell ADJ phone) cellphone cell-phone PDA wireless)	USPAT	OR	ON	2005/04/23 23:21
S14	12	(image WITH size) AND (character\$3 WITH count\$3 WITH line\$1) AND (@ad < "20011503" @rlad < "20011503") AND ((cell ADJ phone) cellphone cell-phone PDA wireless)	USPAT	OR	ON	2005/04/23 23:26
S15	273	(image WITH size) AND (character\$3 WITH count\$3 WITH line\$1) AND (@ad < "20011503" @rlad < "20011503")	USPAT	OR	ON	2005/04/23 23:26

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L4	18	("4653112" "4686332" "4774655" "5007085" "5175854" "5224060" "5309564" "5313578" "5329619" "5471318" "5471615" "5519851" "5561446" "5577177" "5579481" "5583978" "5592657" "5630168"). PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2005/05/15 15:23
L7	3	("6750830", "6691281", "6765596"). pn.	USPAT	OR	OFF	2005/05/15 15:26
S1	534	file adj2 conversion	USPAT	OR	OFF	2005/05/04 00:11
S2	16	(file adj2 conversion) and (handheld or PIM or limited-capability)	USPAT	OR	OFF	2004/07/22 22:14
S3	6	(file adj2 conversion) and (handheld or PIM or limited-capability) and service\$2 and icon	USPAT	OR	OFF	2005/05/15 15:26
S4	13	(large adj2 display) same shap\$3 same (small adj2 display)	USPAT	OR	OFF	2004/07/22 22:03
S6	1	(file adj2 conversion) same (point of interest) and(handheld or mobile or PDA or PIM)	USPAT	OR	OFF	2004/07/22 22:12
S7	40	(file adj2 conversion) same (point of interest)	USPAT	OR	OFF	2004/07/22 22:12
S8	16	(file adj2 conversion) and (handheld or PIM or limited-capability-display)	USPAT	OR	OFF	2004/07/22 22:14
S9	50	(file adj2 conversion) and (handheld or PIM or limited-capability-display or PDA or mobile)	USPAT	OR	OFF	2004/07/22 22:14
S11	1	09/809668	US-PGPUB; USPAT	OR	OFF	2005/05/04 00:13
S12	1	S11 AND count	US-PGPUB; USPAT	OR	OFF	2005/05/04 00:15
S13	1	(character\$1 NEAR2 (count\$3 number\$1)) WITH line WITH text WITH shap\$3	US-PGPUB; USPAT	OR	OFF	2005/05/04 00:17
S14	309	(character\$1 NEAR2 (count\$3 number\$1)) WITH line WITH text	US-PGPUB; USPAT	OR	OFF	2005/05/04 00:18
S15	235	(character\$1 NEAR2 (count\$3 number\$1)) WITH line WITH text	USPAT	OR	OFF	2005/05/04 00:20
S16	27	((character\$1 NEAR2 (count\$3 number\$1)) WITH line WITH text) AND (image\$1 NEAR3 size\$1)	USPAT	OR	OFF	2005/05/04 00:21
S17	42	((character\$1 NEAR2 (count\$3 number\$1)) WITH line WITH text) AND (image\$1 NEAR3 size\$1)	US-PGPUB; USPAT	OR	OFF	2005/05/04 00:39

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L4	18	("4653112" "4686332" "4774655" "5007085" "5175854" "5224060" "5309564" "5313578" "5329619" "5471318" "5471615" "5519851" "5561446" "5577177" "5579481" "5583978" "5592657" "5630168"). PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2005/05/15 15:23
L7	3	("6750830", "6691281", "6765596"). pn.	USPAT	OR	OFF	2005/05/15 15:26
S1	534	file adj2 conversion	USPAT	OR	OFF	2005/05/04 00:11
S3	6	(file adj2 conversion) and (handheld or PIM or limited-capability) and service\$2 and icon	USPAT	OR	OFF	2005/05/15 15:26
S11	1	09/809668	US-PGPUB; USPAT	OR	OFF	2005/05/04 00:13
S12	1	S11 AND count	US-PGPUB; USPAT	OR	OFF	2005/05/04 00:15
S13	1	(character\$1 NEAR2 (count\$3 number\$1)) WITH line WITH text WITH shap\$3	US-PGPUB; USPAT	OR	OFF	2005/05/04 00:17
S14	309	(character\$1 NEAR2 (count\$3 number\$1)) WITH line WITH text	US-PGPUB; USPAT	OR	OFF	2005/05/04 00:18
S15	235	(character\$1 NEAR2 (count\$3 number\$1)) WITH line WITH text	USPAT	OR	OFF	2005/05/04 00:20
S16	27	((character\$1 NEAR2 (count\$3 number\$1)) WITH line WITH text) AND (image\$1 NEAR3 size\$1)	USPAT	OR	OFF	2005/05/04 00:21
S17	42	((character\$1 NEAR2 (count\$3 number\$1)) WITH line WITH text) AND (image\$1 NEAR3 size\$1)	US-PGPUB; USPAT	OR	OFF	2005/05/04 00:39



US006765596B2

(12) **United States Patent**
Lecton et al.

(10) Patent No.: **US 6,765,596 B2**
(45) Date of Patent: **Jul. 20, 2004**

(54) **MULTI-FUNCTIONAL APPLICATION
LAUNCHER WITH INTEGRATED STATUS**

(75) Inventors: **David B. Lecton**, Raleigh, NC (US);
Mark E. Molander, Cary, NC (US);
John L. Scanlon, Raleigh, NC (US)

(73) Assignee: **International Business Machines
Corporation**, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 523 days.

(21) Appl. No.: **09/794,641**

(22) Filed: **Feb. 27, 2001**

(65) **Prior Publication Data**

US 2002/0140742 A1 Oct. 3, 2002

(51) Int. Cl.⁷ **G09G 5/00**

(52) U.S. Cl. **345/835; 345/764; 345/772**

(58) Field of Search **345/864, 865,
345/866, 700, 702, 762-765, 835, 810,
761, 767, 771, 772; 707/1, 102**

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,845,644	A	7/1989	Anthias et al.	364/521
5,497,455	A *	3/1996	Suga et al.	345/835
5,570,109	A *	10/1996	Jenson	345/823
5,745,110	A *	4/1998	Ertemalp	345/764
5,995,940	A *	11/1999	Ramaley	705/9
6,366,898	B2 *	4/2002	Taivalsaari et al.	707/1
6,385,662	B1 *	5/2002	Moon et al.	719/318
6,417,874	B2 *	7/2002	Bodnar	345/854

OTHER PUBLICATIONS

"SilverScreen by PocketSensei", subtitled "the world's most
powerful software in the palm of your hand, POCKET-
SENSEI SilverScreen 1.0", Internet-published article

printed Feb. 15, 2001 (publication date unknown), <<http://www.pocketsensei.com/>>, 2 pages.

"SilverScreen Feature List", subtitled "the world's most
powerful software in the palm of your hand, POCKET-
SENSEI Feature List", Internet-published article printed
Feb. 15, 2001 (publication date unknown), <<http://www.pocketsensei.com/features.htm>>, 2 pages.

Billard E. et al: "A GUI for a manager of Lightweight
Processes" Software Engineering Notes, association for
Computing Machinery. New York. vol. 20, No. 5, Dec. 1,
1995 pp. 48-50. Page 17, line 42 -p. 18, line 18 fig 2.

Friedmann M: "Tools for Windows NT performance Moni-
toring" vol. 24, No. 12, Nov. 12, 1996, pp. 17 - 30.

Mei-Ling Hsu et al: "Building Soft Real-Time Monitors
Based on Software Reuse", Engineering of complex Com-
puter systems, 1998. ICECCS '98. Proceedings Fourth IEEE
International Conference on Monterey, CA, Aug. 10-14, 98,
Los Alamitos, CA.

* cited by examiner

Primary Examiner—John Cabeca

Assistant Examiner—Tadesse Hailu

(74) *Attorney, Agent, or Firm*—Jeanine S. Ray-Yarlettis;
Marcia L. Doubet; Gregory M. Doudnikoff

(57) **ABSTRACT**

User interfaces, methods, systems, and computer program
products for improving interactions with users of pervasive
computing devices such as personal digital assistants, Web-
enabled cellular phones, Web appliances, wearable comput-
ing devices, so-called "smart" appliances in the home, and
so forth. A multi-functional application launcher is defined
that is specifically adapted for use on devices with limited
display space and which use pen-based input or other similar
input means. This application launcher enables improved
navigation, and provides an improved way to show task
status information to the device user. Multiple functions per
task are supported, and status information is provided, all
from an application launcher view.

38 Claims, 20 Drawing Sheets

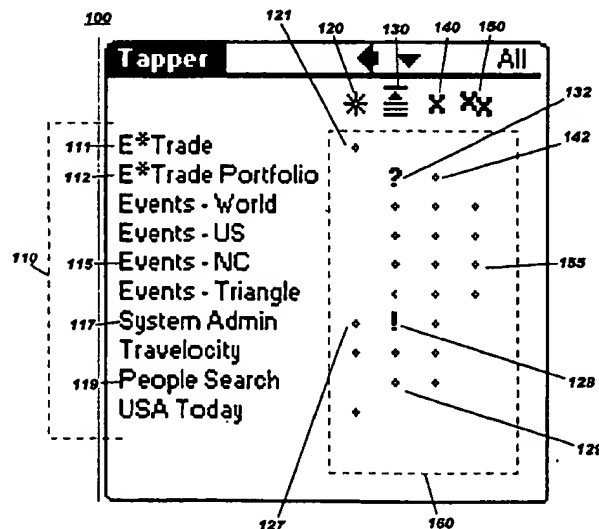


FIG. 1A

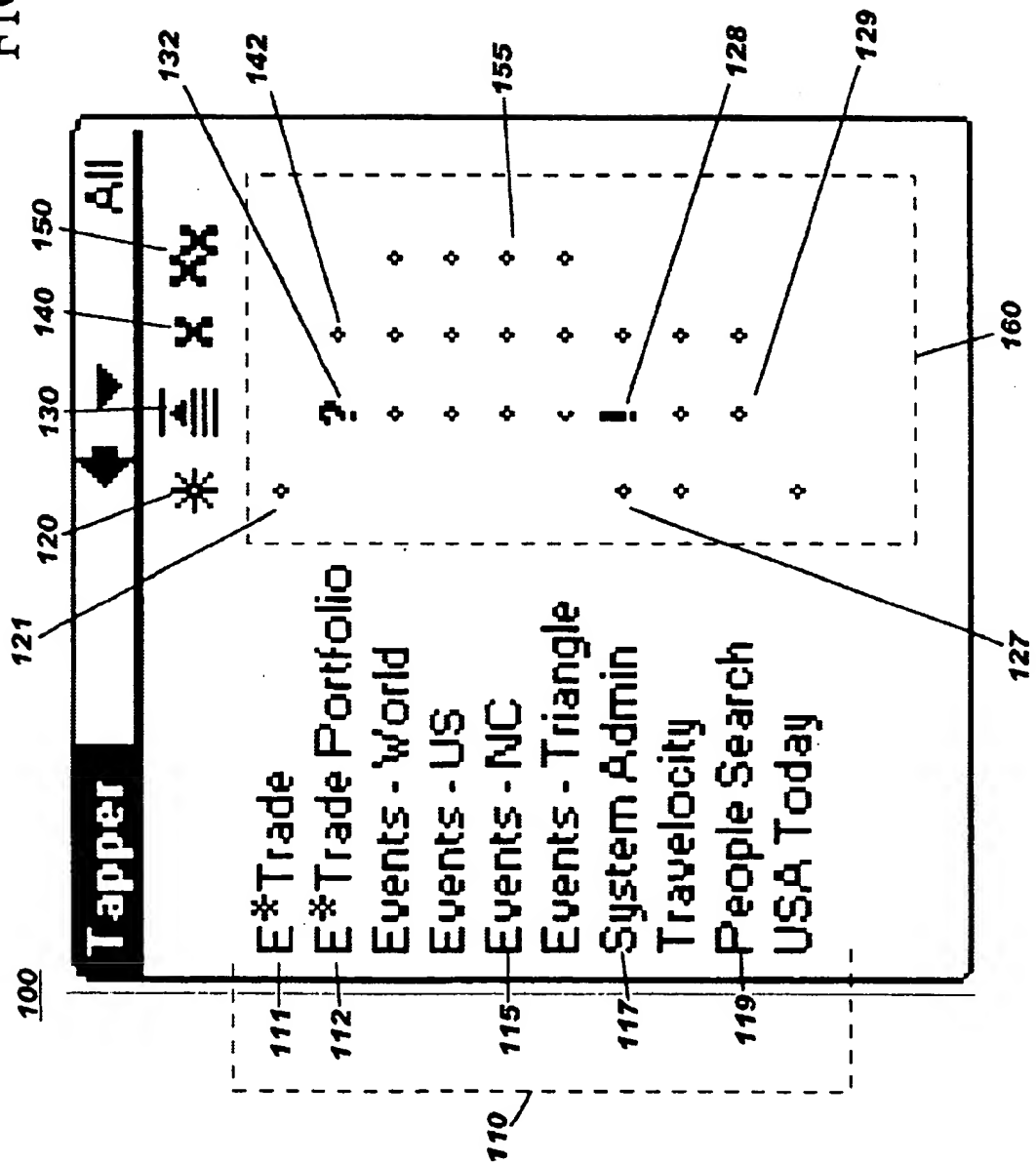


FIG. 1B

100

Tapper		All	
		*	≡
E*Trade	◇		XXX
E*Trade Portfolio		?	◇
Events - World		◇	◇
Events - US		◇	◇
Events - NC		◇	◇
Events - Triangle		◇	◇
System Admin	◇	!	◇
Travelocity	◇	◇	◇
People Search		◇	◇
USA Today	◇		

FIG. 1C

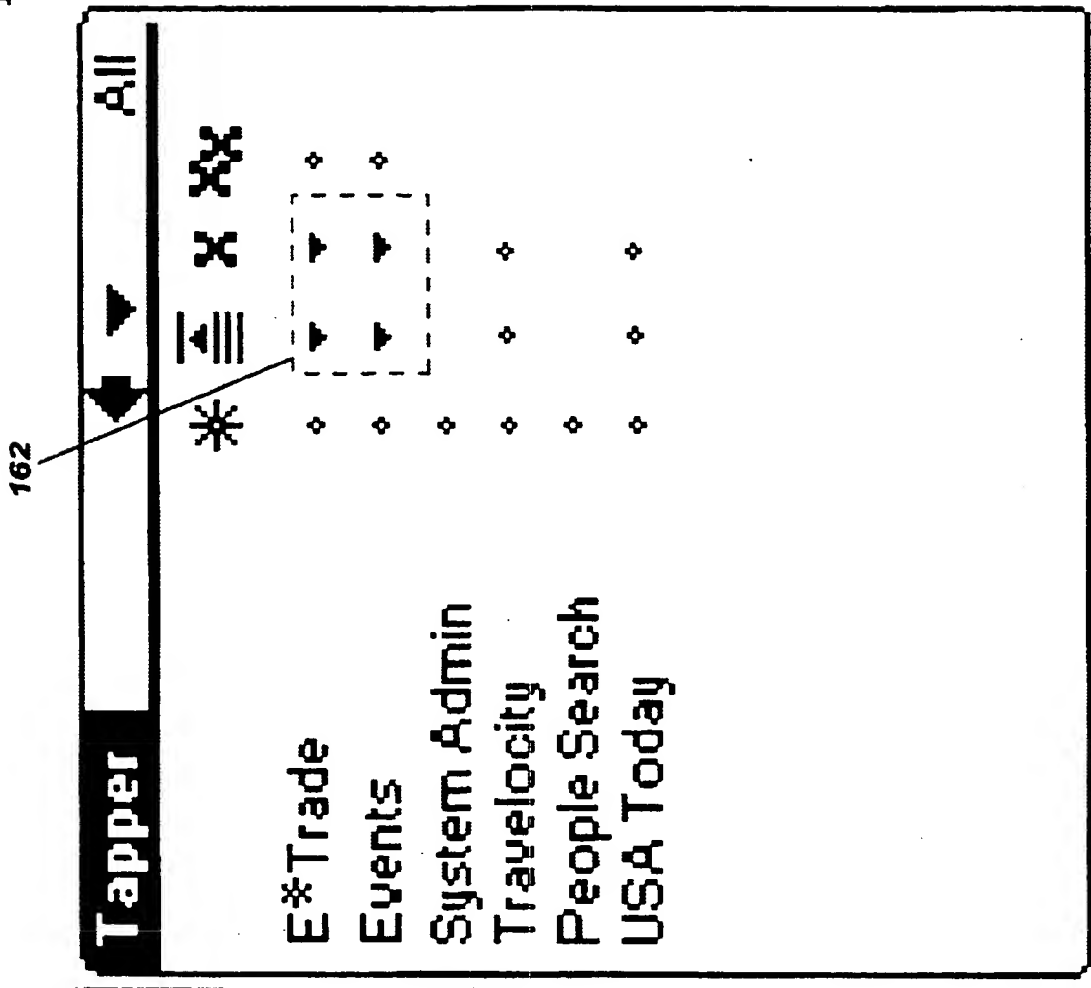


FIG. 1D

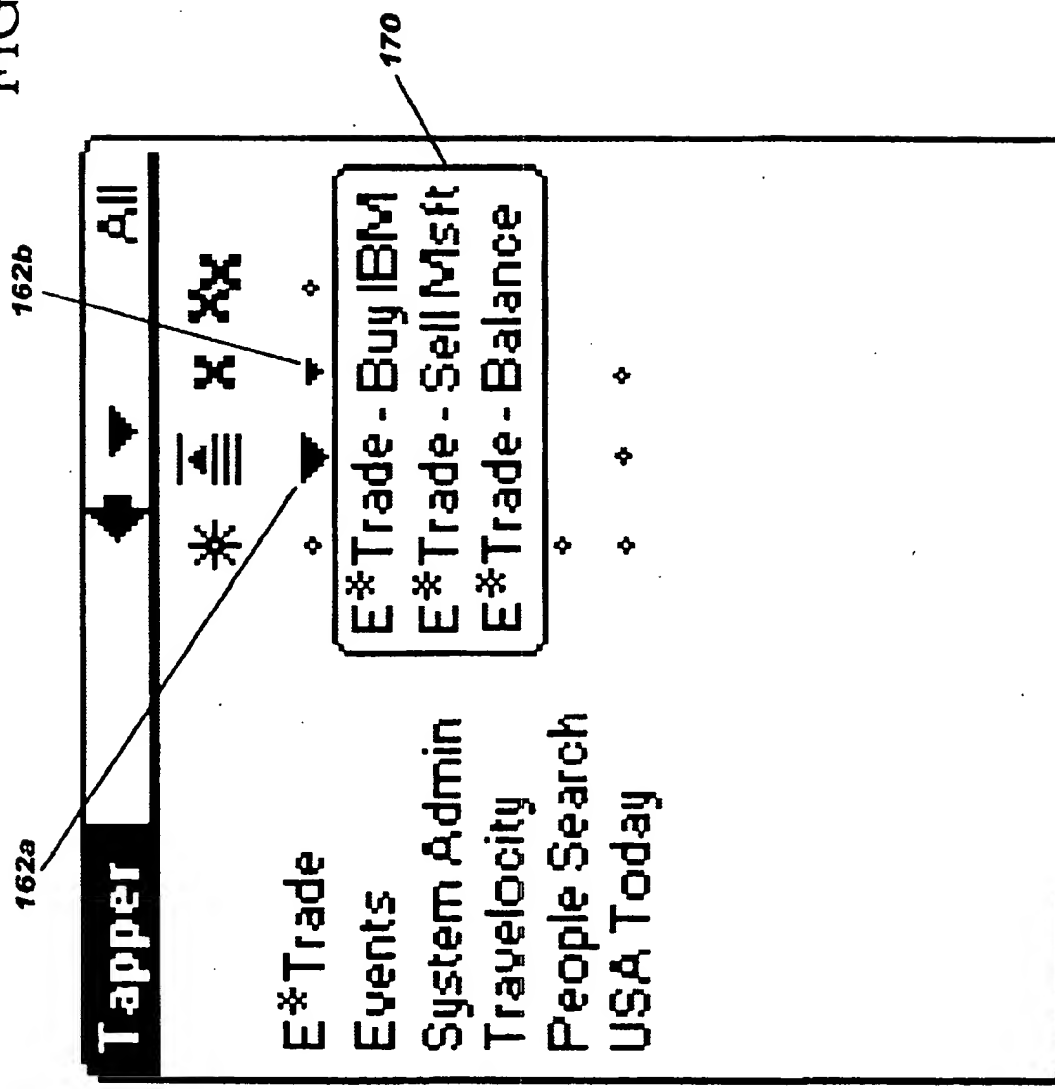


FIG. 1E

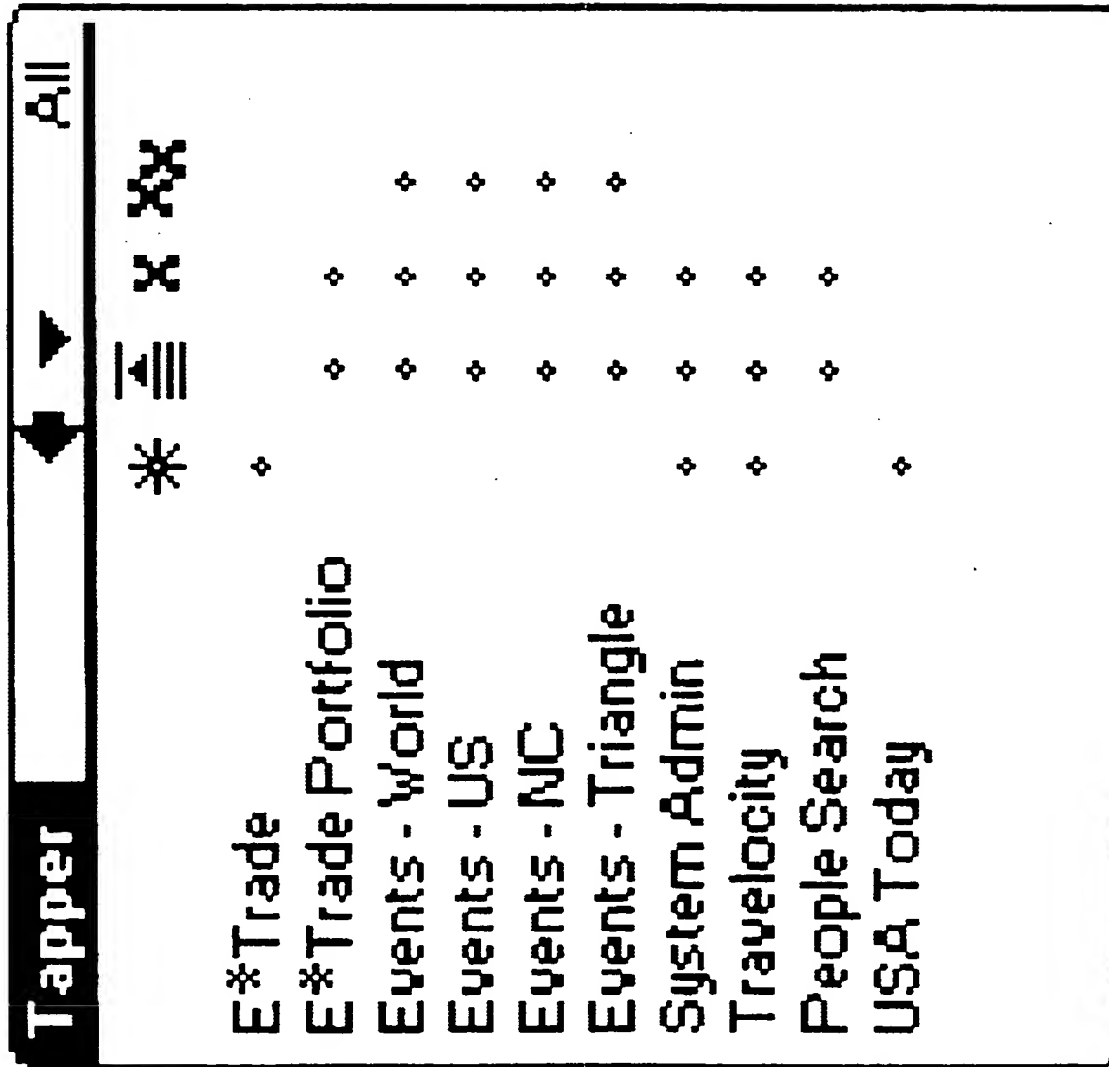


FIG. 2A

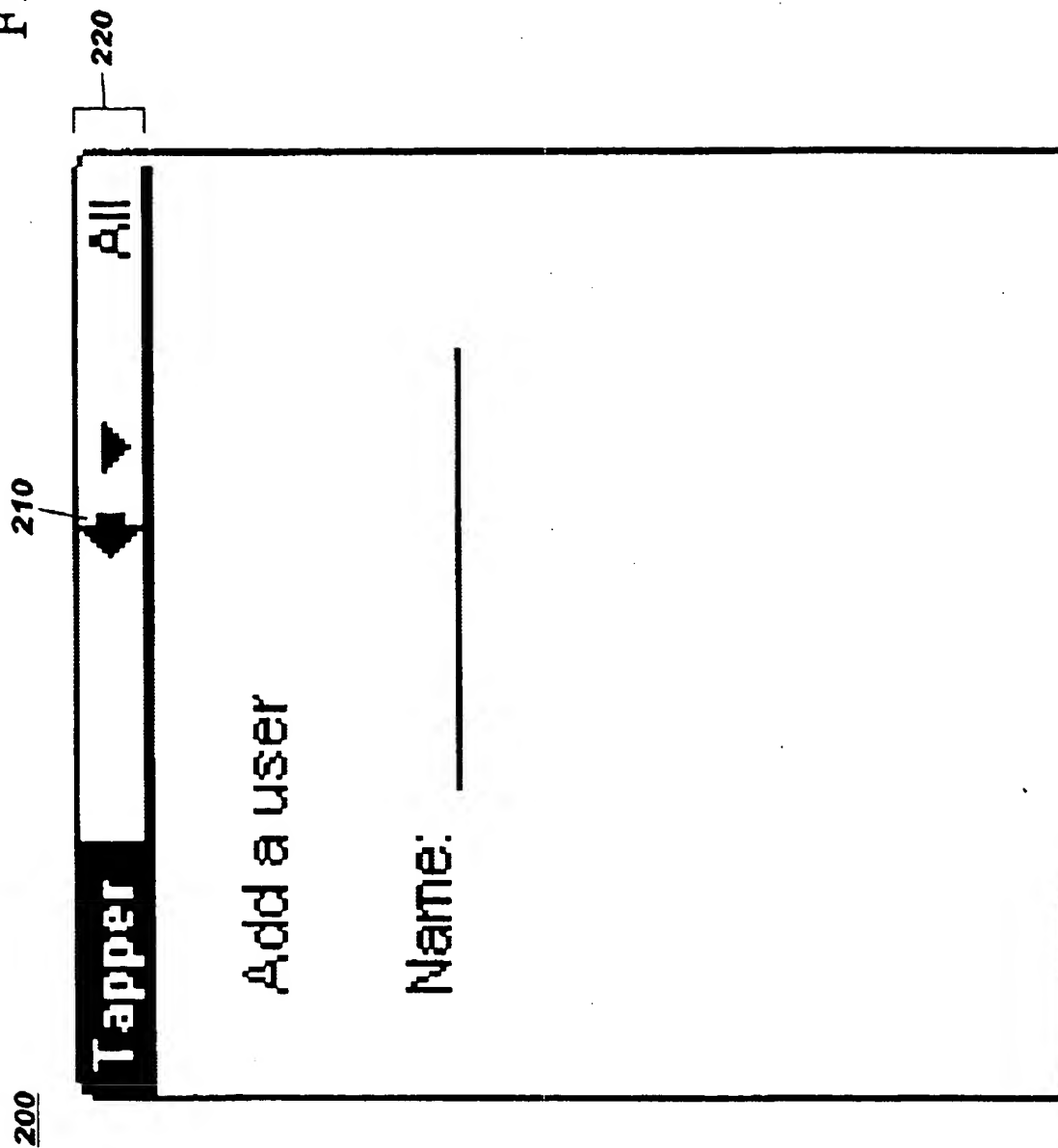


FIG. 2B

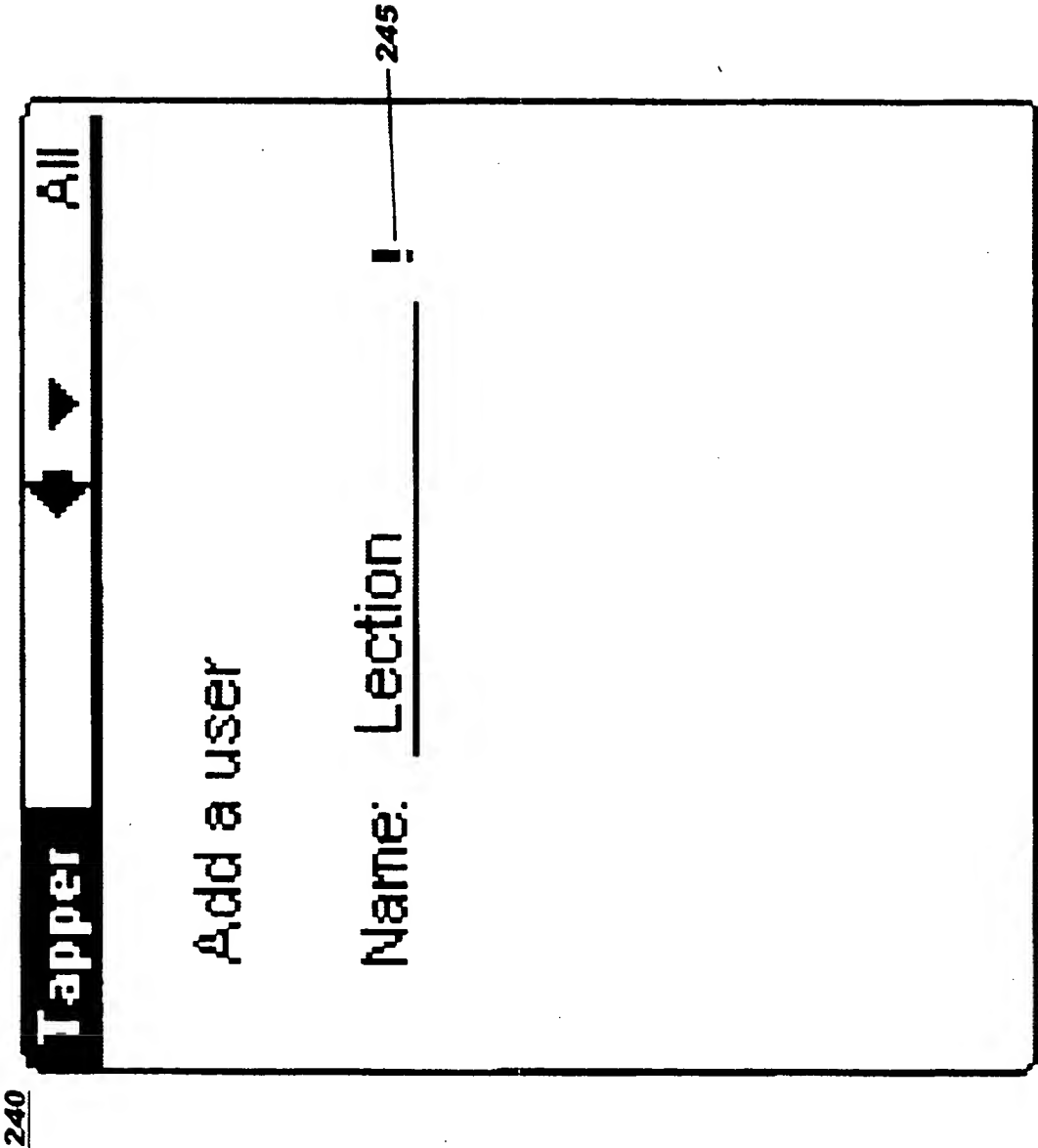


FIG. 2C

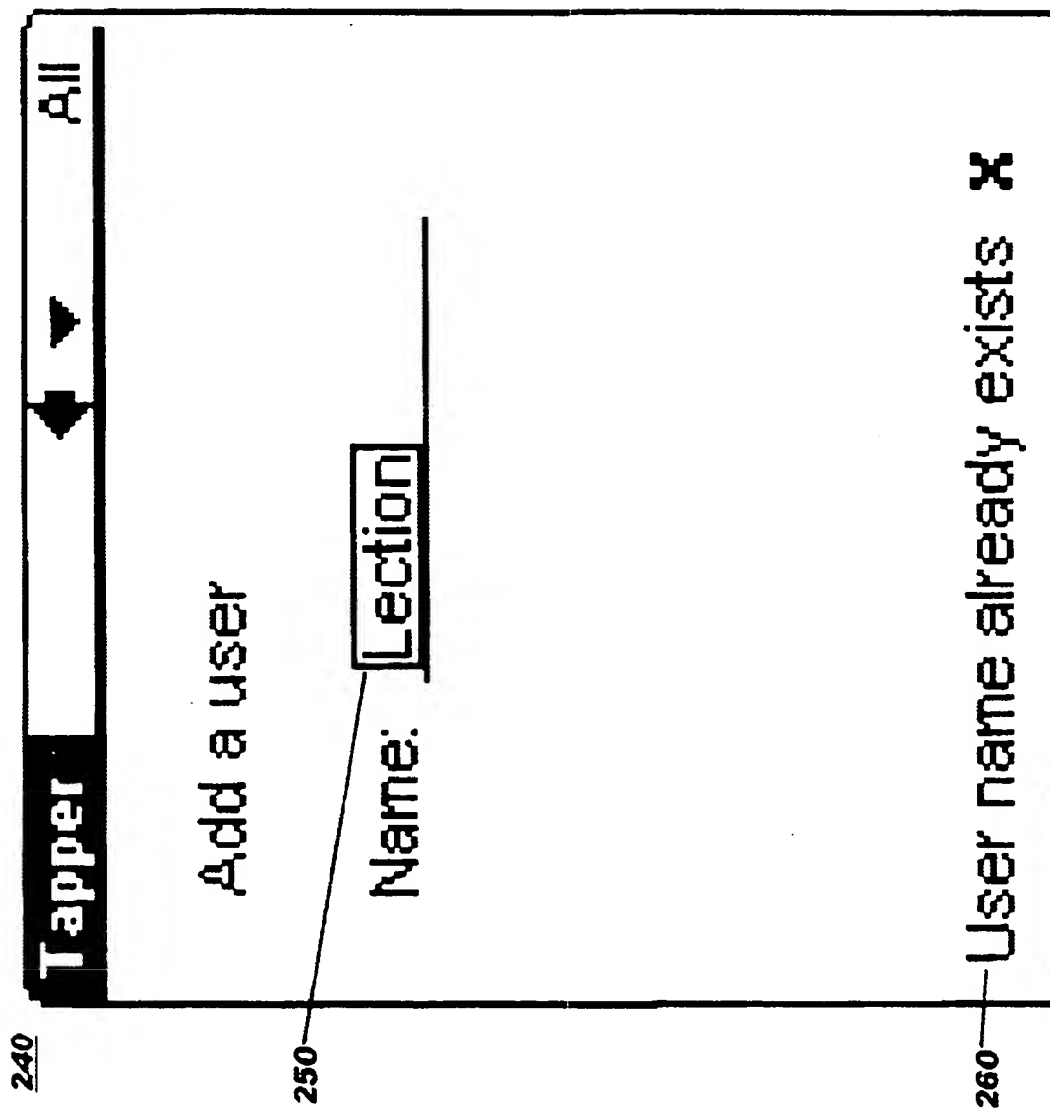


FIG. 3

300 Start Task

- * Start a new instance of this task
- E Event notification has arrived to start this task

310 Running Task

- * Bring selected task of this name to surface of task viewer
- ? Task is requesting user input
- ! Task has information to show to user

320 Stop Task

- * Close selected instance of this task after surfacing view
- ! Task completed with some condition. After surfacing view, terminate.
- C Task completed. Close task without surfacing view.

330 Stop All Tasks

- * Close all instances of this task

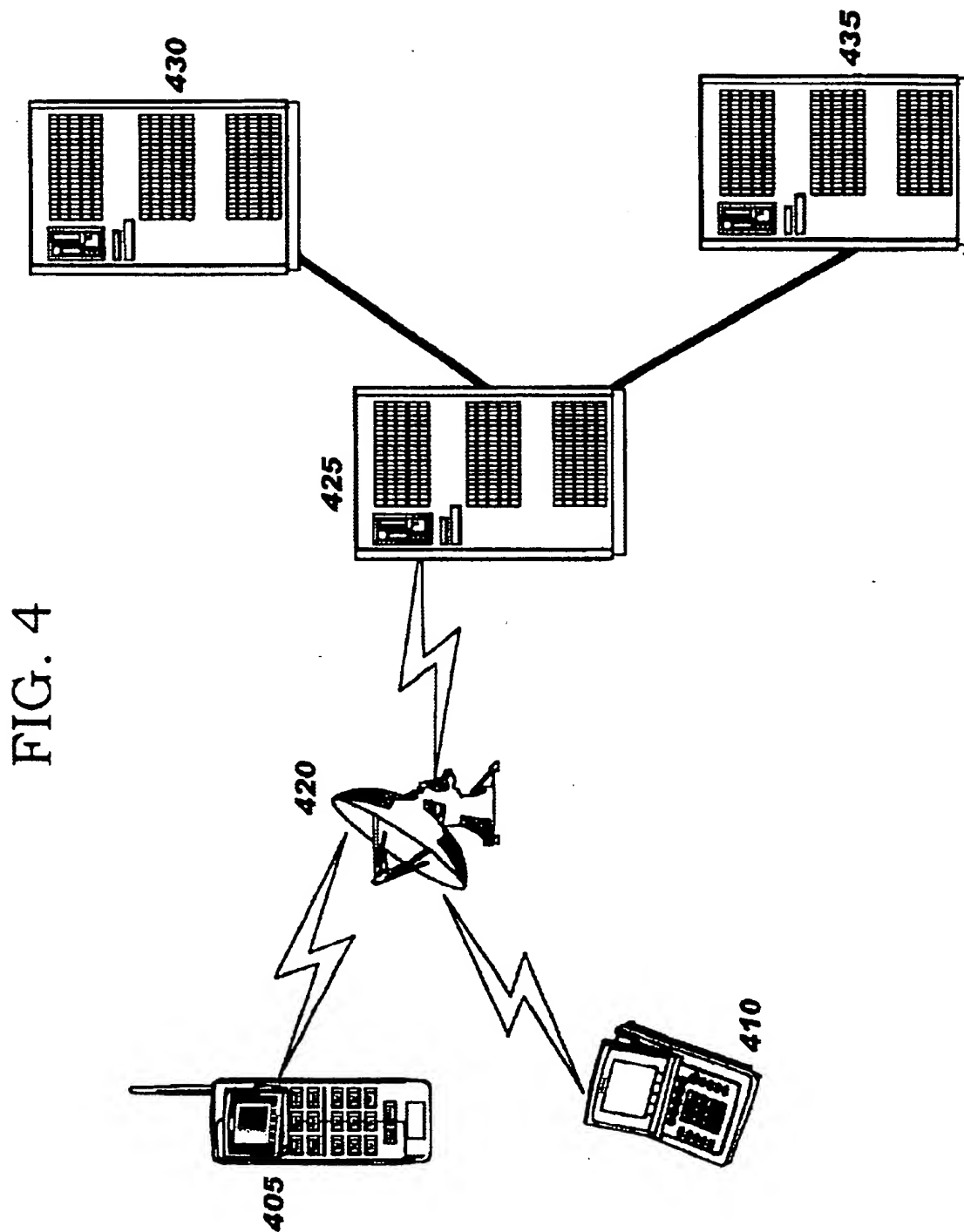


FIG. 5

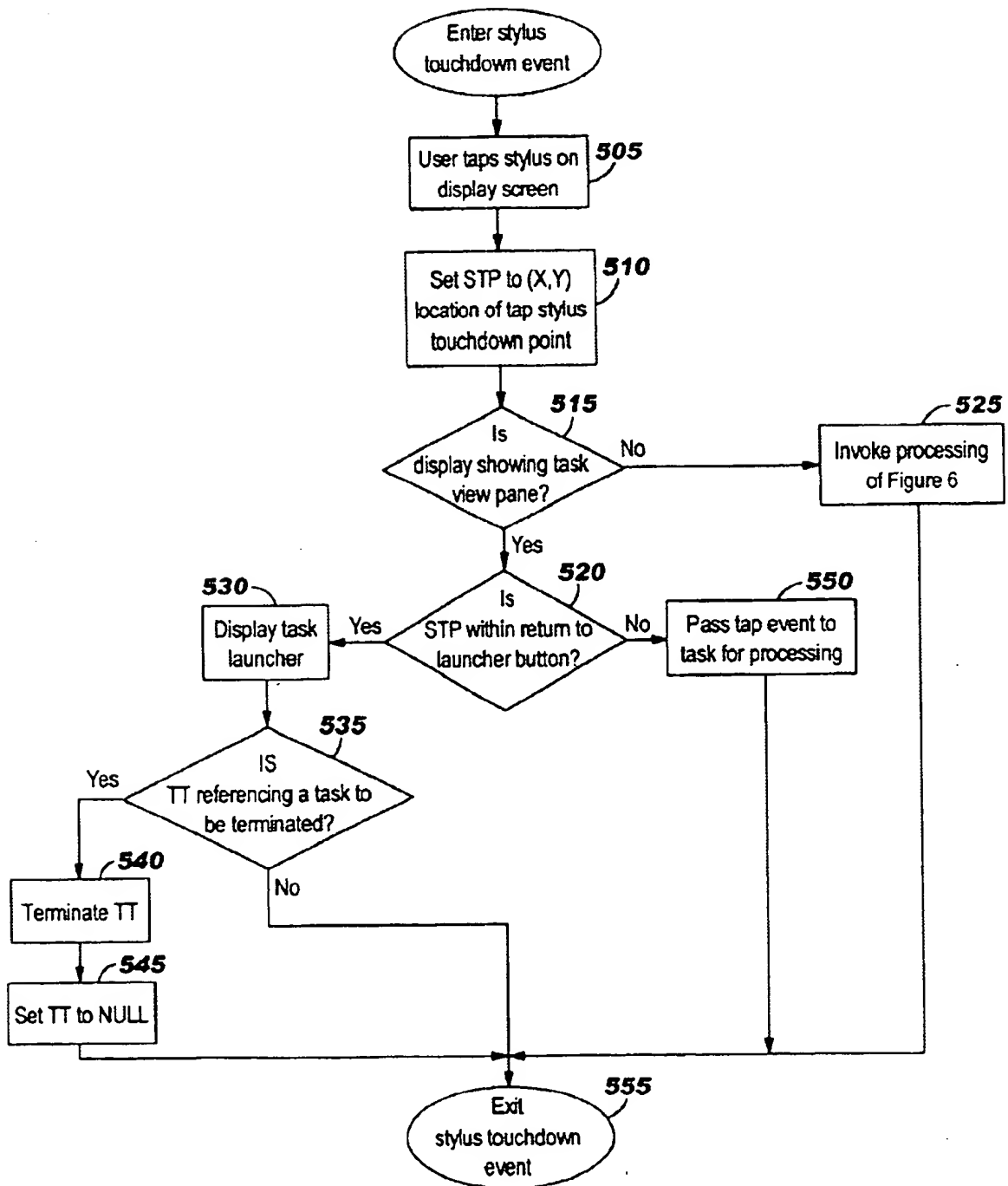


FIG. 6

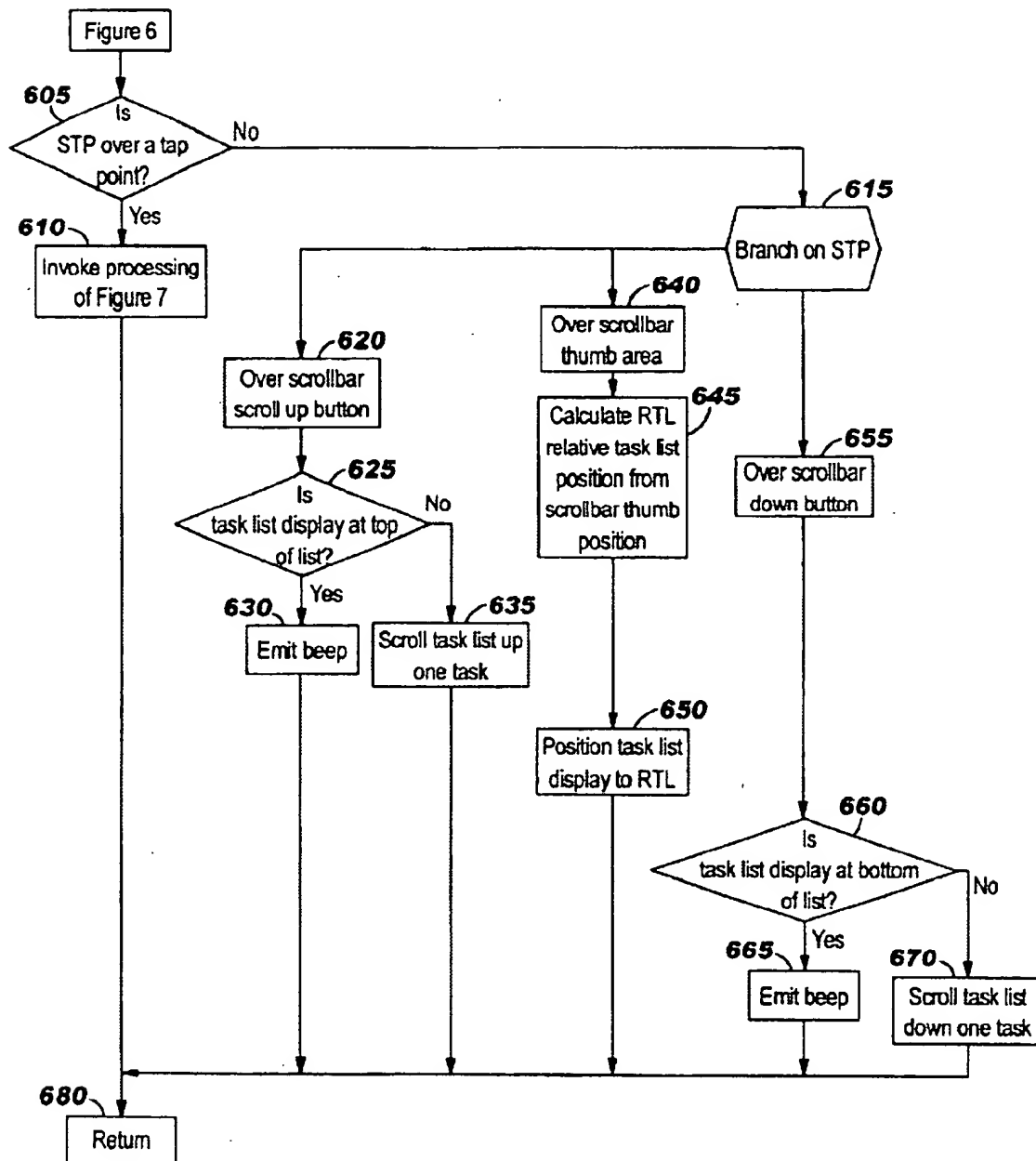


FIG. 7

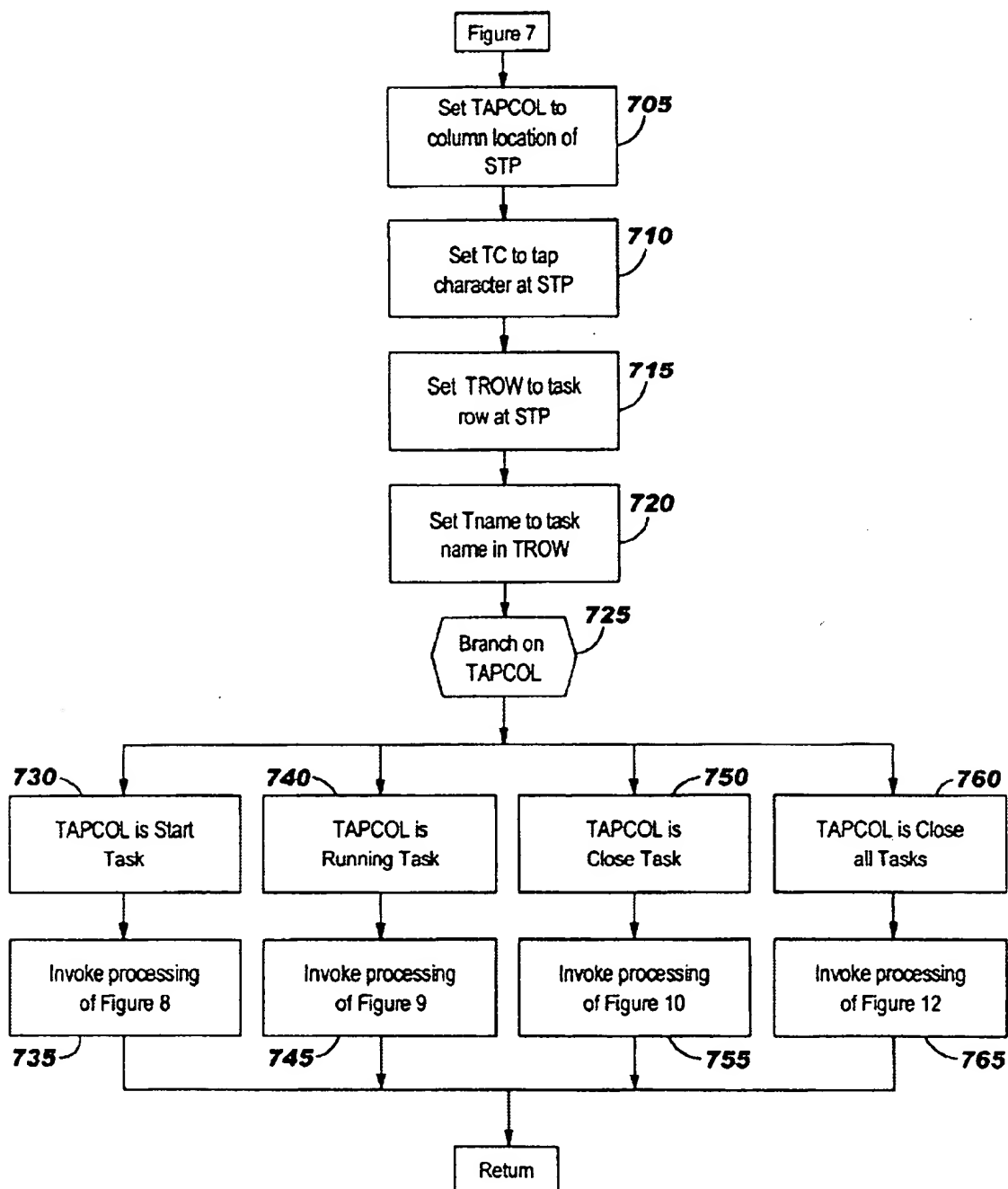


FIG. 8

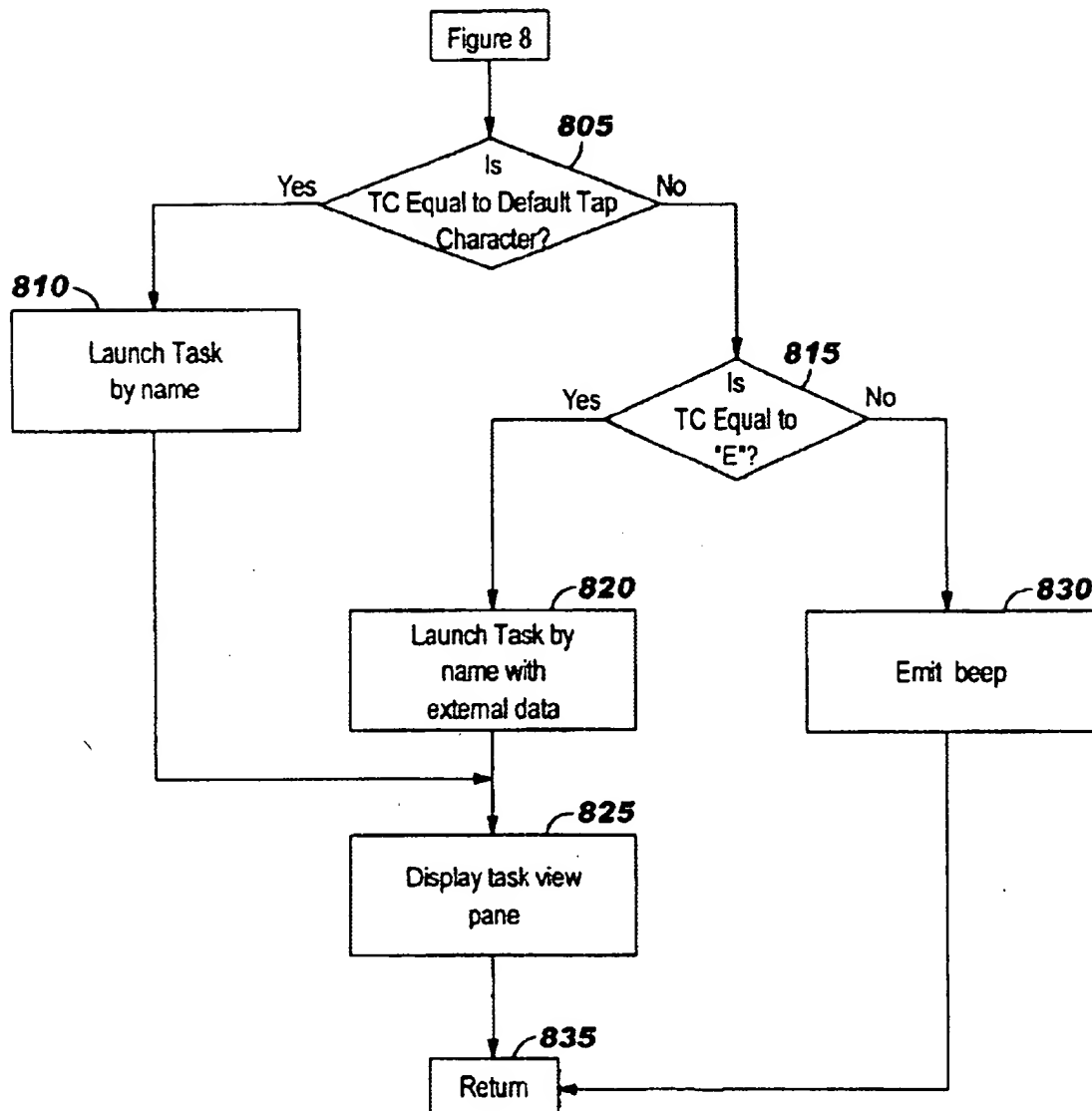


FIG. 9

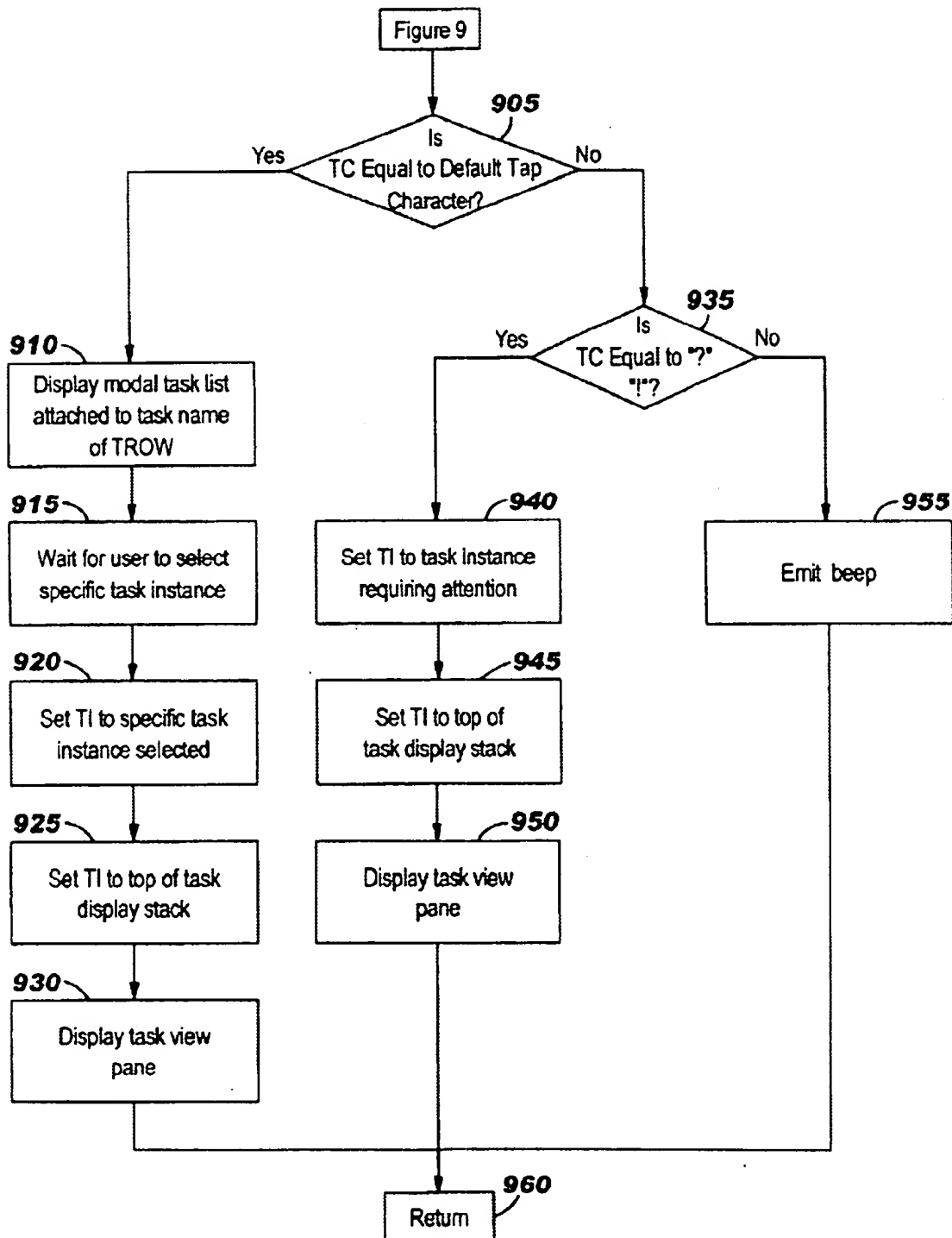


FIG. 10

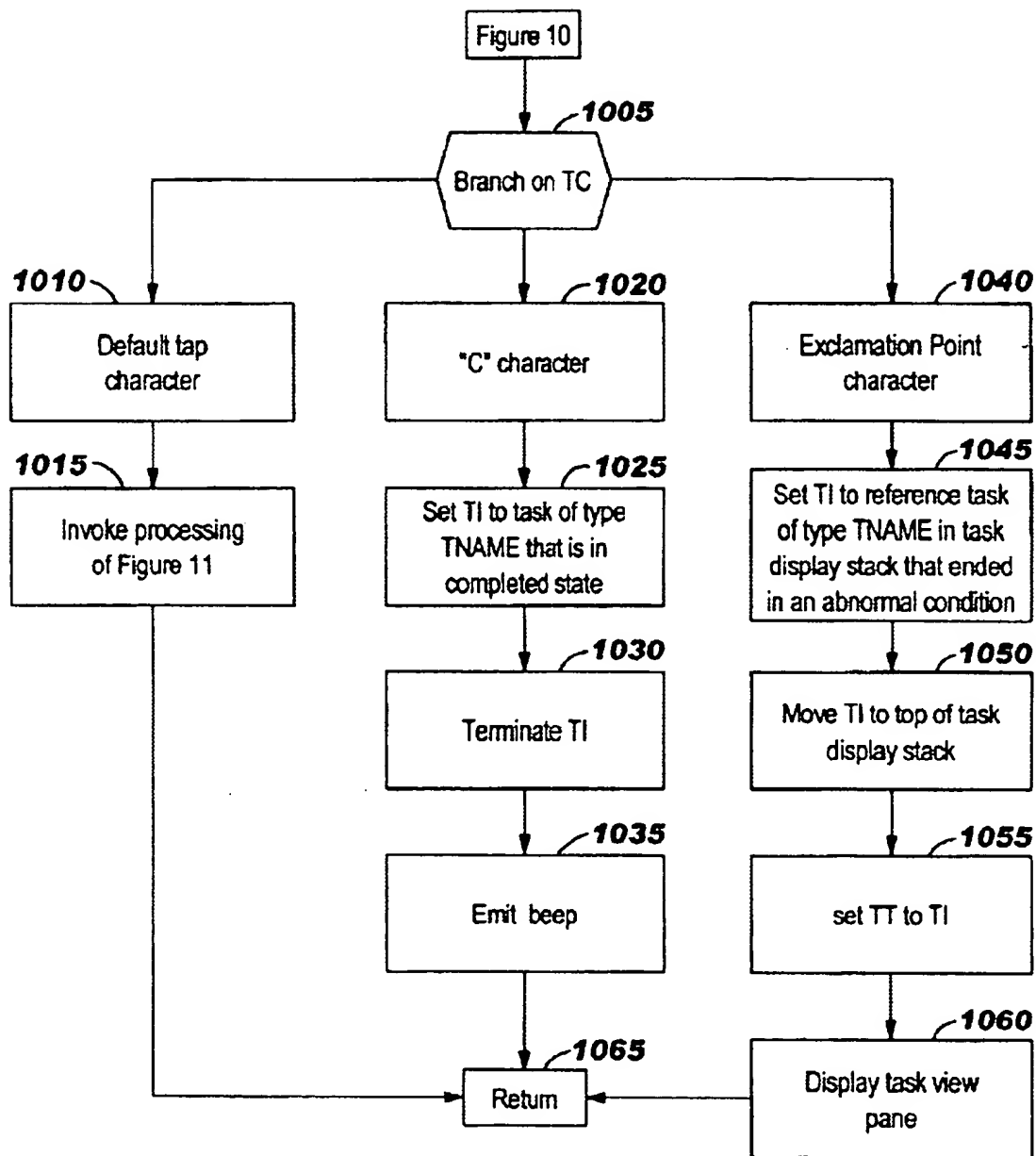


FIG. 11

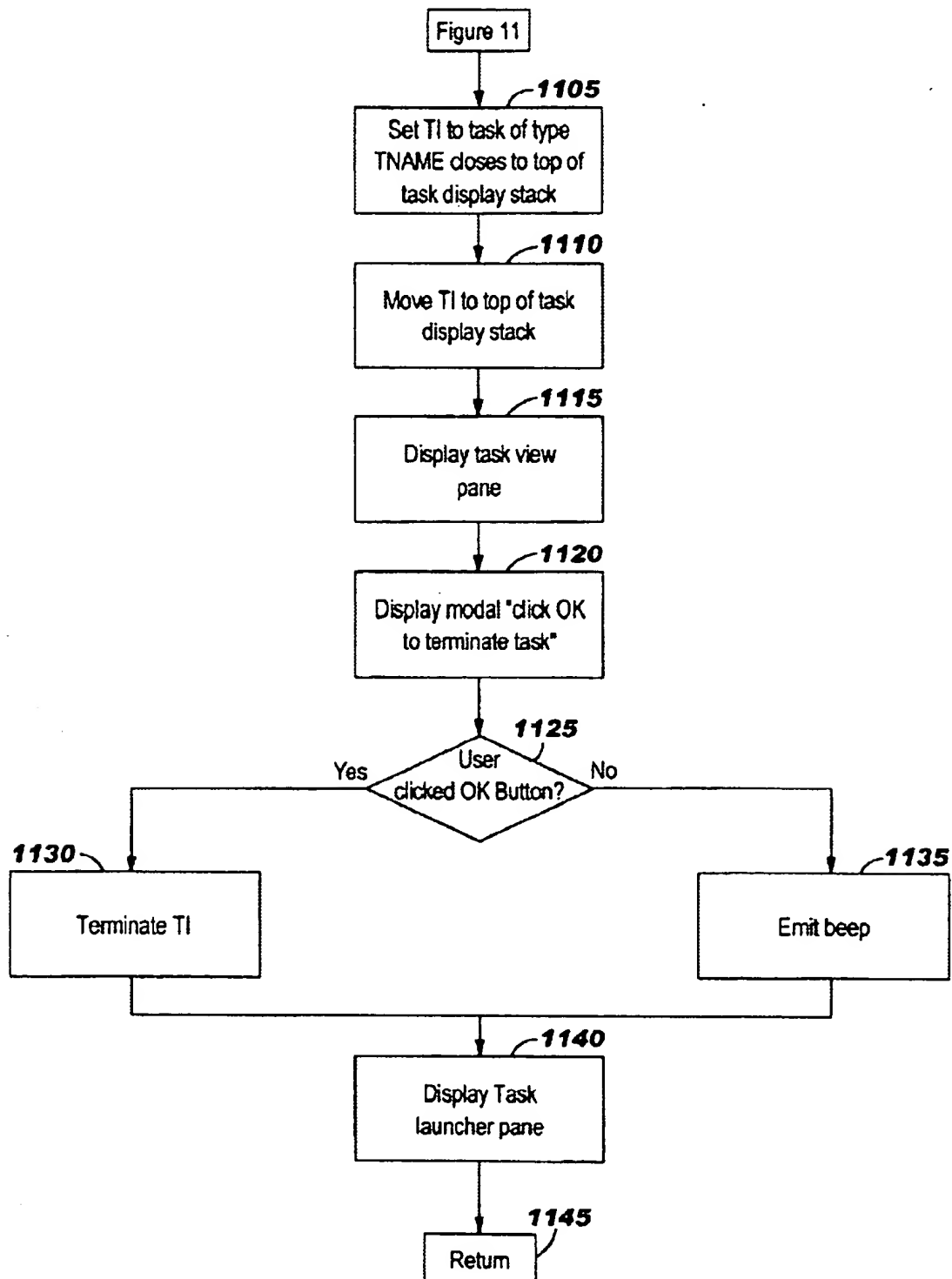


FIG. 12

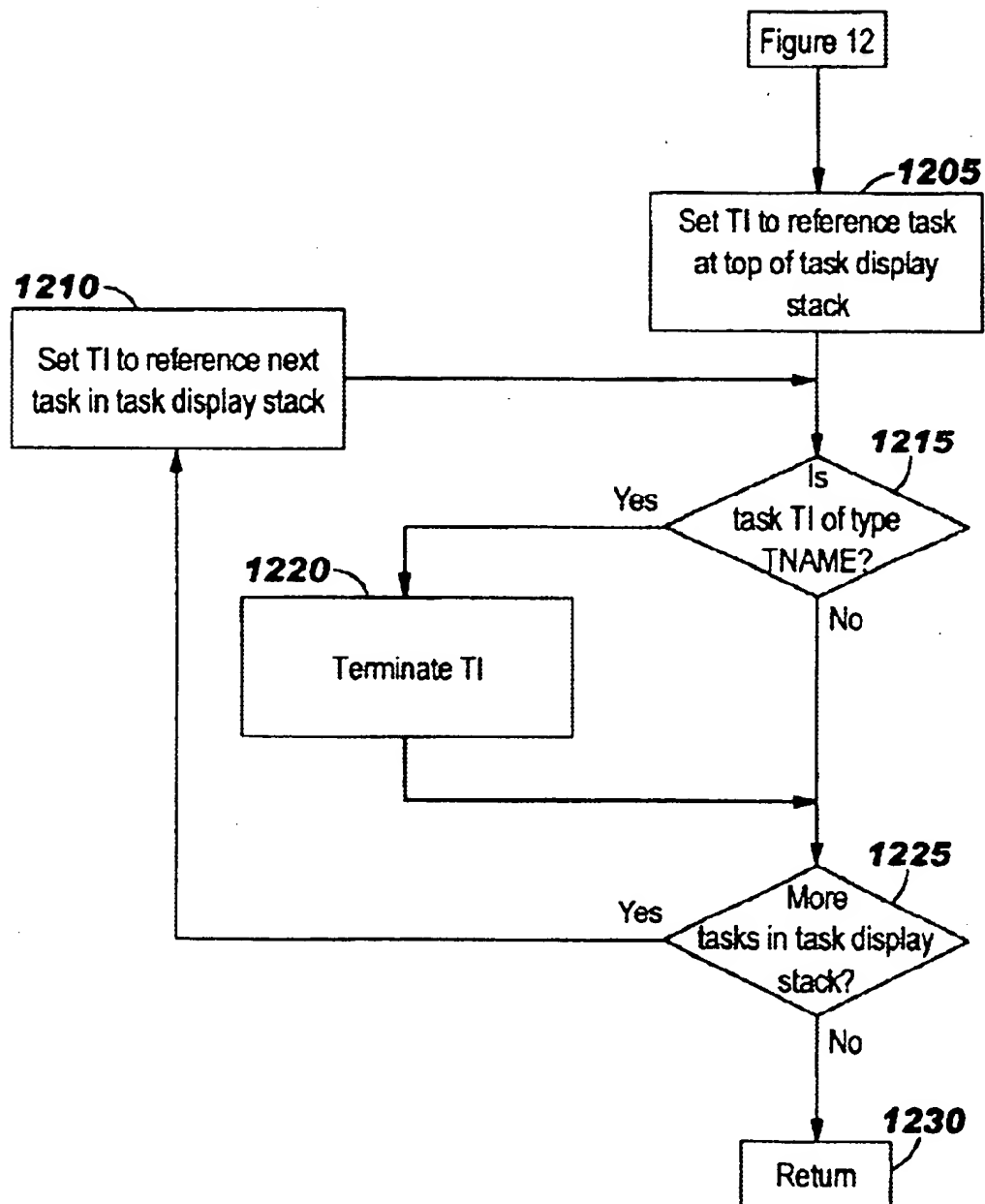


FIG. 13A

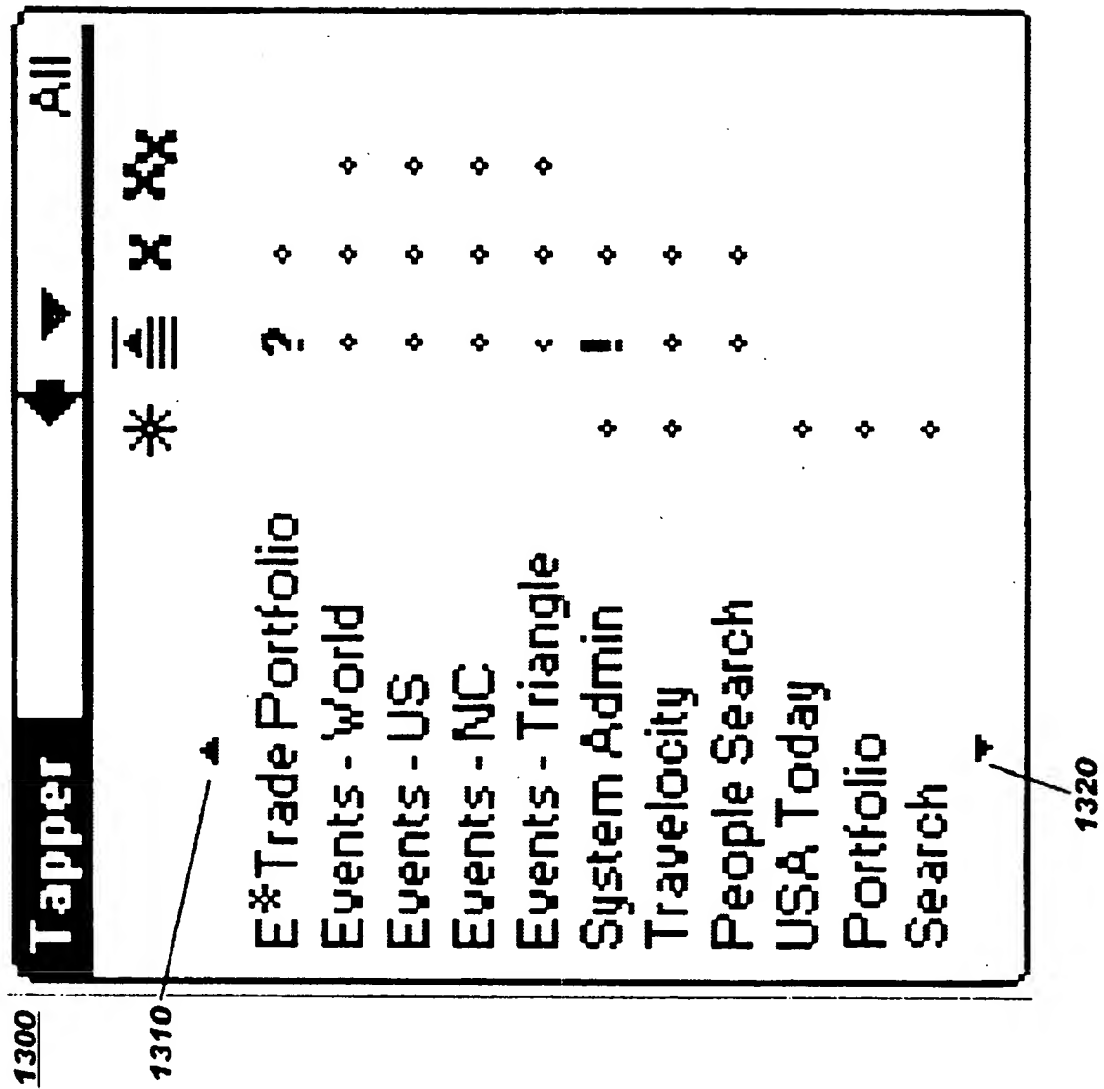
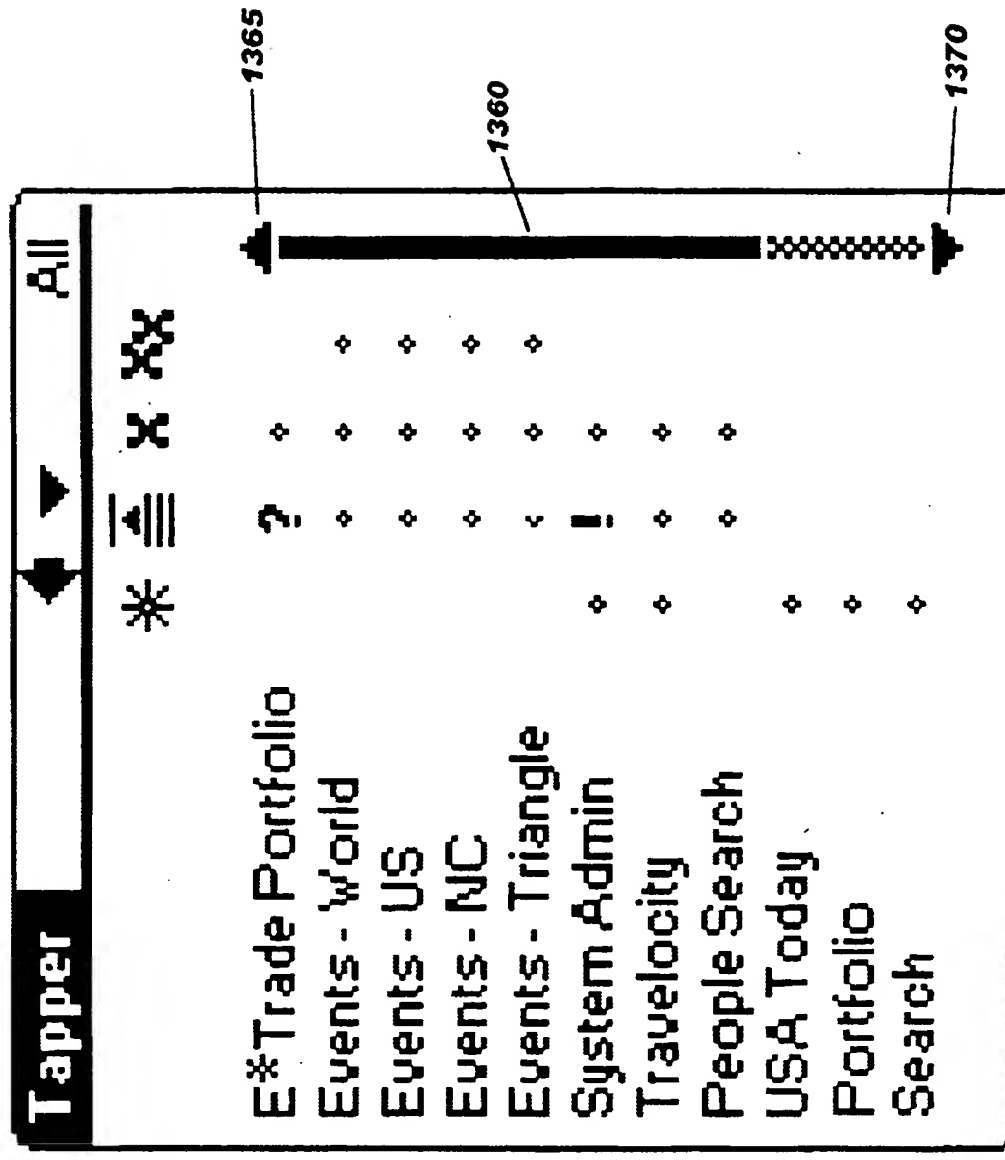


FIG. 13B



MULTI-FUNCTIONAL APPLICATION LAUNCHER WITH INTEGRATED STATUS

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to pervasive computing, and deals more particularly with user interfaces, methods, systems, and computer program products for improving a user's interactions with pervasive computing devices by providing an application launcher that is specifically adapted to such devices.

2. Description of the Related Art

Pervasive computing devices are the most rapidly growing class of computing devices available today. Such devices include personal digital assistants ("PDAs"), Web-enabled cellular phones, Web appliances, wearable computing devices, so-called "smart" appliances in the home, and so forth. It is predicted that by the year 2004, pervasive computing devices will be more widely used than personal computers. The reasons for this are evident. The Internet and World Wide Web (hereinafter, "Internet") bring mobility, and mobility dictates a need for portability. As more and more people incorporate these devices into their everyday lives, the cost of such devices becomes more affordable. Now factor in the coming of inexpensive wireless access to the Internet, and it is easy to understand why these pervasive computing devices are becoming ever more popular. (Note: reference herein to "computing" devices is also intended to include devices used primarily for communications or other purposes, such as cellular phones.)

Pervasive computing devices vary in their characteristics. Often, pervasive computing devices are adapted to operation while being held, and many accept user input with a touch-sensitive display screen or via a light pen or other type of stylus. Typically, the display screen on these devices is much more limited in size than the displays available on personal computers. The user interface models that have been developed in recent years to optimize a user's interactions with a more traditional type of computing device therefore do not necessarily adapt well to use with pervasive computing devices. The Palm class of devices appears to have a market leadership position for handheld pervasive computing devices with respect to popularity and usability. However, even within this class of devices there are several drawbacks. For example,

A single application launcher paradigm has not yet emerged. Most often, the application launcher used on such devices is very similar to a Windows program manager. In this approach, applications are typically reflected on a workspace using icons. With the constrained viewing space available on most pervasive computing devices, an icon-based display often proves to be inefficient and cumbersome. For example, it may take a considerable amount of time to search for an icon throughout a series of display screens. The SilverScreen® application launcher offers some improvements, for example by providing the ability to drag and drop applications and data to services (such as dragging an application to the trash can to delete it), but is still oriented toward use of application icons and resembles a (very small) Windows desktop. ("SilverScreen" is a registered trademark of PocketSensei.)

Existing application launchers are not optimized for use with a pen or stylus interface. (Hereinafter, the words "pen" and "stylus" are used interchangeably.) They still require the user to search for an application based upon its iconic

representation, and then tap the icon to start execution of the application. These existing launchers have often implemented good pen support given their paradigm, but are not specifically adapted nor optimized for a pen-based interface.

The user interfaces of these devices are not optimized for switching from a launched application or task to the application launcher and back. Note that many devices in the Palm class provide a "home" button to enable switching back to the application launcher from a task view: this button would not be needed if a uniform way existed to switch between the launcher and a launched application or task (referred to hereinafter as a "task" for ease of reference).

Accordingly, what is needed are improvements to the user interface of these devices that address the drawbacks of existing solutions.

SUMMARY OF THE INVENTION

An object of the present invention is to provide improvements to the user interface of pervasive computing devices that address the drawbacks of existing solutions.

Another object of the present invention is to provide a user interface for pervasive computing devices that supports improved navigation.

Another object of the present invention is to provide a user interface that is optimized for a pen-based interface.

A further object of the present invention is to provide a user interface that is optimized for handheld devices.

Still another object of the present invention is to provide a user interface that is specifically adapted for devices having a limited display area.

Yet another object of the present invention is to provide a user interface for pervasive computing devices that enables quickly and efficiently switching between an application launcher and a task view.

Other objects and advantages of the present invention will be set forth in part in the description and in the drawings which follow and, in part, will be obvious from the description or may be learned by practice of the invention.

To achieve the foregoing objects, and in accordance with the purpose of the invention as broadly described herein, in a first aspect the present invention provides a user interface for pervasive computing devices. In one aspect, this user interface comprises a graphical representation of a collection of one or more tasks of the pervasive computing device, and a display area corresponding to the graphical representation of the collection, wherein the display area indicates status information for each of the one or more tasks and also provides accessibility to a plurality of functions for the collection of tasks.

In another aspect, this comprises a multi-functional application launcher view and a plurality of task views, wherein a user of the pervasive computing device selectively navigates between the multi-functional application launcher view and selected ones of the task views. The multi-functional application launcher view comprises: a displayed list of tasks available to the user for execution by the pervasive computing device; and a displayed grid corresponding to the displayed list of tasks, wherein the displayed grid comprises a plurality of columns and a row for each task in the displayed task list. The displayed grid preferably further comprises entries at intersections of the columns and the rows, wherein the entries provide status information for the tasks in the displayed list and enable performing a plurality of actions (e.g. launching tasks) relative to selected ones of the tasks in the displayed list.

The status information preferably includes which tasks are executing, which tasks are requesting input from the user, and which tasks have information to display to the user. The status information may also include a notification that a particular task should be started by the user. In this latter case, the notification may be received because of an event which occurred in another task or perhaps because a message arrived for the user, in which case the message indicates that the particular task should be started. The message may optionally include input values to be used by the particular task when it is started. The input values may be encoded within a structured markup language document or other means.

The status information may also include which tasks have completed normally, and/or which tasks have completed abnormally. In the latter case, the user may see the task view of a selected one of the tasks that have completed abnormally by activating the entry providing the status information of the selected task.

The actions that may be performed preferably include starting execution of a selected task and surfacing the task view for a selected task, and may also include stopping execution of a selected task. Preferably, entries in the grid may be selectively activated by the user to perform the actions. The activation may comprise tapping a selected entry with a stylus or pen input device. The actions may also include stopping execution of a selected group of tasks, in which case the selected group may comprise multiple instances of a particular task from the displayed list.

A graphical selection area may be provided on the task view for a selected task where this selection area may be activated by the user to return to the multi-function application launcher view.

In yet another aspect, the present invention provides methods, systems, and computer program products for providing an improved user interface for pervasive computing devices. This aspect preferably comprises displaying a multi-functional application launcher view, and enabling a user of the pervasive computing device to selectively navigate between the multi-functional application launcher view and selected ones of a plurality of task views. Displaying the multi-functional application launcher view preferably comprises displaying a list of tasks available to the user for execution by the pervasive computing device, and displaying a grid corresponding to the displayed list of tasks, wherein the displayed grid comprises a plurality of columns and a row for each task in the displayed task list. The columns preferably correspond to life cycle points of the displayed list of tasks, and this aspect may further comprise displaying entries at selected intersections of the columns and the rows, wherein the entries provide status information about particular life cycle points of the task associated with that row and/or graphical depictions of available actions for particular life cycle points of the task associated with that row. Optionally, the user may be allowed to modify the life cycle points and/or to modify the graphical depictions of available actions. The graphical depictions may be activated to perform actions for particular life cycle points of the task associated with that row. A selected displayed entry may be revised when the task associated with that row has different status information or when the task associated with that row has a different available action. In this case, an application programming interface invocation may be received which indicates the different status or available action.

Optionally, a graphical indication may be displayed on the multi-functional application launcher view when the dis-

played list of tasks exceeds a display capacity of the pervasive computing device.

The actions in this aspect preferably include starting execution of a selected task and surfacing the task view for a selected task, and may also include stopping execution of a selected task. The status information preferably includes which tasks are executing, which tasks are requesting input from the user, and which tasks have information to display to the user. A plurality of instances of a particular task may be executing, in which case surfacing a task view preferably further comprises surfacing the task view of a selected one of the plurality when the user activates the graphical depiction for surfacing the task view of the particular task. The particular one may be, for example, the instance which was most recently viewed by the user. Or, if the activated graphical depiction indicates that input is requested from the user, then the particular one may be the instance which is requesting the input. Or, if the activated graphical depiction indicates that information is available for presenting to the user, then the particular one may be the instance which has the available information. As yet another alternative, the particular one may be selected by the user. In this latter case, the activated graphical depiction preferably indicates presence of the plurality, and a selectable representation of the plurality is presented to the user in response to the activation of the graphical depiction. The particular one is then preferably the instance which is selected by the user from the selectable representation.

The user of the pervasive computing device may selectively navigate between the multi-functional application launcher view and selected task views.

The present invention will now be described with reference to the following drawings, in which like reference numbers denote the same element throughout.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGS. 1A-1E illustrate various aspects of an improved user interface for pervasive computing devices, according to the present invention;

FIGS. 2A-2C depict sample task view displays which illustrate use of an optional feature of the present invention;

FIG. 3 provides a list of command and status symbols that may be supported by an implementation of the present invention;

FIG. 4 illustrates a network in which the present invention may optionally be used;

FIGS. 5-12 provide flowcharts depicting logic that may be used in implementing preferred embodiments of the present invention; and

FIGS. 13A-13B illustrate additional features that may be provided in the improved user interface of the present invention.

DESCRIPTION OF PREFERRED EMBODIMENTS

The present invention provides improvements to the user interface of pervasive computing devices. A multi-functional application launcher is defined that is specifically adapted for use on devices with limited display space and which use pen-based input or other similar input means. This application launcher enables improved navigation, and provides an improved way to show task status information to the device user. In particular, the multi-function application launcher of the present invention reflects task status information on the same view which supports launching tasks. The advantages

5

of this new user interface will become clear from the following description of the novel features of the present invention.

An exemplary embodiment of the improved user interface of the present invention is illustrated in FIGS. 1A–1E. (The content of FIGS. 1A and 1B is identical, except that the reference numbers which are used to assist in describing the application launcher appear in FIG. 1A but have been removed from FIG. 1B in order to more clearly illustrate the appearance of this novel user interface technique.) The application launcher of preferred embodiments is structured as a list of task names and a corresponding grid of information pertaining to a set of life cycle points for the tasks. The life cycle points preferably correspond to (1) the ability to start a task; (2) a current execution status of a task; and (3) the ability to stop a task. (In preferred embodiments, when a task is “stopped” by the user it is cancelled, although in alternative embodiments the stop function may be implemented as merely pausing task execution.) In some implementations, the present invention may be used with a multi-tasking operating system, such as Windows CE or Epoch32/Psion. In these implementations, an additional life cycle point may correspond to the ability to stop a group of executing tasks. Optionally, embodiments of the present invention may be adapted for allowing a user to add more life cycle points and/or to delete one or more of the provided life cycle points (although the start task and execution status life cycle points should be defined as basic functionality which is not capable of deletion).

A sample list of task names is shown at element 110 of FIG. 1A. The life cycle points are represented by columns 120, 130, 140, 150. To save display space, the function of each column is preferably depicted by a column heading which uses an iconic symbol or a limited number of representative characters. For purposes of illustration, a star symbol (*) is used for the Start Task column 120; a switch or stack symbol (see element 130 in FIG. 1A) is used for the Running Task column 130; an “X” is used for the Stop Task column 140; and an “X_x” symbol is used for the Stop All Tasks column 150. The grid 160 comprises a row for each task name from the list 110, where each row has an intersection with the column for each life cycle point. At each point of intersection, the grid may either be empty or it may contain a “tap point”. As used herein, the term “tap point” refers to a small area of the display screen in which a graphical indicator is provided, where that graphical indicator may be tapped by the user with a pen to trigger a particular action. If there is no graphical indicator at an intersection point, then tapping there has no effect (although optionally, an indication such as an audible beep may be provided to signal the user). The presence of tap points at particular intersection points also provides certain status information to the user, as will be described in more detail herein. For purposes of illustration, a “dot” symbol is used herein as a default tap point symbol, as shown at 121.

By defining the user interface of the present invention with a grid approach, multiple functions may be performed for a particular task from this single view, providing significant usability improvements over the prior art user interface approach for pervasive computing devices.

As an example of how the tap points in grid 160 are used by the present invention to convey status information and to provide an improved pen-based interface, the presence of a tap point symbol 121 at the intersection of the row for the “E*Trade” task 111 and the Start Task column 120 indicates that this task is available to be started. Thus, if the user taps on this particular tap point, it will trigger execution of the

6

E*Trade task. In this example, task 111 has not yet been started, and therefore there is no tap point for its row in any of the remaining columns 130, 140, or 150. On the other hand, the E*Trade Portfolio task 112 has a tap point 132 in the Running Task column 130 and another tap point 142 in the Stop Task column 140, indicating that this task is now running (but may be stopped if the user so desires). The absence of a tap point in the Start Task column 120 for this row indicates that the task 112 is not available to be started.

Tapping on a tap point from the Running Task column 130 causes the application launcher view 100 to be replaced with a task view corresponding to the task in the row of that tap point. For example, if tap point 129 is tapped, the application launcher will be replaced by a view for the “People Search” task 119.

In a basic embodiment of the present invention, only a single tap point symbol (i.e. the default symbol) is supported. (See FIG. 1E for a depiction of an example application launcher which only supports this basic symbol.) In a more advanced embodiment, tap points other than the default symbol are supported. These other symbols are preferably used to convey additional status information. For example, the tap point shown at 132 is depicted as a question mark “?”. This tap point may be used in an implementation to indicate to the user that an underlying task is requesting user input. The tap point shown at 128 is depicted as an exclamation point “!”. This symbol may be used to indicate that an underlying task needs to show the user some type of status information, such as an error message.

As an example of how the “?” may be used, suppose the E*Trade Portfolio task 112 is creating portfolio information and needs to know for whom this information is to be created. The presence of tap point 132 notifies the user that the task is awaiting user input. When the user taps on tap point 132, the application launcher will be replaced with a view pane for corresponding task 112. For example, a task view pane such as that illustrated as element 200 of FIG. 2A may be displayed. This example task view pane also illustrates an advantageous navigational mechanism provided by the improved user interface of the present invention, wherein a user may tap on the left-pointing arrow 210 (or other analogous symbol) to immediately return to the application launcher view 100 from a currently-displayed task pane view. Preferably, this return function is provided from all task view panes. By placing this arrow in the title bar (shown as element 220) of the task view panes, very little additional display space is consumed.

An example of using the “!” tap point is illustrated in FIGS. 2B and 2C. By tapping on tap point 128 for the System Admin task 117, the corresponding task view pane is then automatically surfaced. For purposes of illustration, suppose this task view pane is as shown at 240 in FIG. 2B. In preferred embodiments, the surfaced task view pane includes this same “!” symbol in proximity to the item to which the user’s attention should be directed. In this example, the “!” symbol is placed at the end of an input field (see element 245) used for entering a user name, thereby conveying that there is some type of additional information available about this user name or perhaps about this input field. Upon tapping on the “!” in this pane, that information is preferably added to the task view pane, as shown at element 260 in FIG. 2C. In this case, the information is an error message pertaining to the input value “Lecction”. The entered input value associated with this error message may optionally be graphically highlighted in some manner, as illustrated at element 250 in FIG. 2C by surrounding the input value with a box. Preferably, the status information

displayed at 260 includes a symbol that may be used to indicate acknowledgement of the message (which is an "X" in preferred embodiments). When the user is finished viewing this status information, he taps on this acknowledgement symbol, and in preferred embodiments, the task view pane is then automatically replaced by the application launcher pane. (In alternative embodiments, the status information may simply be removed from the task view pane, and the user may return to the application launcher pane at his convenience by tapping the left-facing "Return" arrow symbol. Furthermore, in alternative embodiments the status message may already be displayed upon surfacing the task view pane, such that the user is not required to tap on element 245 to see the message.)

In some implementations (e.g. those which do not support multi-tasking), it may be desirable to define the Start Task and Running Task columns 120, 130 as mutually exclusive, such that a tap point may appear in only one of these columns for a particular row at any given time. Typically, when a tap point is shown in the Running Task column 130, a tap point for that row is also displayed in the Stop Task column 140.

In preferred embodiments, tapping on the default tap point from the Stop Task column 140 causes a task view pane for the corresponding task to be surfaced, so that the user may see any information that may be presented therein prior to actually stopping the task's execution. Optionally, the user may also be allowed to confirm whether the task is to be stopped or not. (In alternative embodiments, the task may be stopped without surfacing the task view pane, if desired.) After the task is stopped, preferred embodiments may remove the tap points in columns 130 and 140 of this row and then display a tap point in column 120, indicating that the task has become available for (re-)starting. (Alternatively, the Start Task tap point may be displayed upon receiving an event or other similar notification from the underlying task, indicating that the task is available to be started.)

In advanced embodiments, the Stop Task column 140 may support use of the "!" symbol and/or a "C" symbol as tap points. Preferably, the "!" symbol is used to indicate that a task has ended with some condition that the user should be informed of. By tapping on this symbol, the task view pane will be surfaced. Preferably, an application programming interface ("API") is provided for use by executing tasks, and the task invokes this API to signal that the tap point should be changed to reflect the special tap point symbols discussed herein. The user can then see the information about the particular condition in the context of the task's view pane. Upon returning to the application launcher, the task is purged and the "!" tap point is removed.

A "C" tap point symbol in the Stop Task column preferably indicates that the underlying task completed normally. In such cases, there is typically no additional status information to show to the user, and the task preferably uses the API invocation to request changing the tap point to a "C" rather than to the "!" symbol. Thus, when the user taps on a "C" tap point, preferred embodiments simply terminate the task without surfacing its view pane.

In a multi-tasking environment, more than one task may be executing at the same time, in which case multiple rows of grid 160 will contain tap points in column 130. In addition, more than one instance of a particular task may be executing at the same time. In this latter case, implementations of the present invention preferably provide for surfacing the task view of a predetermined one of these multiple

instances, and optionally for stopping execution for a particular one of the instances, when a tap is received in column 130 or 140, respectively. The predetermined one may be selected in a number of different ways. For example, a record may be kept of which instance was most recently viewed, and that instance may then be selected. Or, the first instance to be started, or perhaps the last instance to be started, may be selected. When a "?" or "!" is displayed as a tap point in the Running Task column 130 for a task with multiple instances, then the instance to be surfaced is preferably the one which generated the event (e.g. the API invocation) causing this special status symbol to replace the symbol which was displayed. As yet another alternative, an explicit selection means may be provided whereby the user determines which instance the switch or stop tap applies to. An example of this latter technique is illustrated in FIG. 1C, where the tap points for the E*Trade task and the Events task are depicted as using downward-pointing arrows (see element 162), for purposes of illustration, to indicate to the user that a list is available for viewing here. FIG. 1D illustrates a result of tapping on the upper left one of these arrow symbols, wherein a pop-up view pane 170 is displayed to show the currently-running instances of the corresponding task. (Note that in this example, the tapped arrow 162a is shown slightly larger than arrow 162b, to emphasize to the user which arrow the pop-up view pane 170 corresponds to.) The user may then select one of these tasks from the list, e.g. by tapping on its displayed name.

In a multi-tasking environment where more than one instance of a particular task may be executing at the same time, implementations of the present invention may optionally provide for stopping execution of all such tasks with a single tap through use of the Stop All Tasks column 150. For example, FIG. 1A indicates that multiple instances of the "Events-NC" task 115 are running by the presence of the tap point 155, and the presence of this tap point also allows the user to stop all of them by touching the pen to the tap point.

A task display stack model is preferably used with the present invention, whereby the view pane at the top of the stack is the one currently displayed. According to preferred embodiments of the present invention, upon tapping on a tap point from the application launcher view, the launcher view 100 is automatically replaced by a task view pane for the task corresponding to the tapped point (as has been briefly discussed above). For example, upon tapping on tap point 128 for the System Admin task 117, the task pane 240 illustrated in FIG. 2B becomes the top entry on the task display stack, and is therefore displayed instead of application launcher 100. There are, however, a limited number of exceptions to this automatic replacement. In preferred embodiments, when a task is closed by selecting the "C" tap point from the Stop Task column or selecting the default tap point from the Stop All Tasks column of the application launcher view, the application launcher view remains on the display. In addition, when the pop-up view pane technique illustrated by FIGS. 1C and 1D is used, the application launcher is preferably only temporarily (and partially) overlaid with the pop-up view pane.

FIG. 3 illustrates the tap point symbols that may be supported in advanced embodiments of the present invention. A particular implementation may support one or more of these special tap point symbols. In addition, an implementation may optionally modify the set of tap point symbols, e.g. to provide one or more additional tap point symbols to account for the unique semantics of particular tasks. It will be obvious to one of ordinary skill in the art how this additional functionality may be provided once the

teachings disclosed herein are known. Furthermore, an optional user input mechanism may be supported whereby the user invokes functions of an API defined for use with the present invention to specify additional user-specific or application-specific tap point symbols and/or to change the provided symbols.

As shown at 300, the tap point symbols supported for the Start Task column of the application launcher are preferably (1) the dot symbol, which starts a new copy of a task and which serves as the default symbol, and (2) an "E", which represents an event occurrence that is being graphically conveyed to the user. The "E" symbol has not been described previously. This symbol may be used, for example, when an action has occurred in some currently-executing task—such as the user pressing a button or other application-specific condition—where a result of this action is that another task should be started. In this case, the task in which the action occurred invokes an API to signify that the currently-displayed tap point in the Start Task column 120 should be replaced with this "E" symbol. As another example, suppose that a systems administrator is responsible for maintaining an up-to-date employee list for a corporation, and that managers of this corporation are responsible for sending e-mail notifications to the systems administrator when new employees are to be added to the list. The systems administrator may have a task on his pervasive computing device which receives these e-mail messages and then automatically updates a tap point symbol in column 120 to reflect that an incoming message has arrived. By monitoring his pervasive computing device for this type of changes to the tap points, the systems administrator is informed of events indicating that the employee list needs to be modified and he can therefore start the necessary task. When this optional event symbol is supported, the associated incoming message or notification may optionally include data to be used as input parameter values for the task for which the event symbol is displayed. For example, suppose that the "Add a user" task shown in FIG. 2B is adapted for this type of automatic event notification and processing. The triggering incoming message may include the employee's name and employee number. One way in which this type of information may be provided is through use of a structured markup language document (such as an Extensible Markup Language, or "XML", document) passed with the incoming message. As an example, the following markup document may be used as input to signify that employee Joe Smith, who is employee number 12345, should be added:

```

<TASK-REQUEST>
  <ADD-USER>
    <USER>
      <LAST-NAME>Smith</LAST-NAME>
      <FIRST-NAME>Joe</FIRST-NAME>
      <EMP-NUM>12345</EMP-NUM>
    </USER>
  </ADD-USER>
</TASK-REQUEST>

```

Upon receiving this markup language document, the tag values may be stripped out and used to prime the task instance that is started when the user taps on the "E" tap point.

Referring again to FIG. 3, element 310 shows the tap point symbols that may be supported for the Running Task column. In preferred advanced embodiments, these comprise (1) the default (e.g. dot) symbol, (2) a question mark (?) symbol, and (3) an exclamation point (!) symbol. As in

element 300, the default symbol is shown as a dot. When a dot symbol in the Running Task column is tapped, this indicates that the corresponding task view pane should be surfaced to the top of the task display stack (thereby replacing the display of the application launcher pane). As stated earlier, the "?" symbol is preferably used to indicate to the user that the corresponding task (or perhaps a particular instance thereof) is requesting user input, and the "!" symbol is preferably used to indicate that the corresponding task (or a particular instance thereof) needs to show the user some type of status information, such as an error message. Preferably, an implementation of the present invention provides support for API invocations from tasks encountering events of these types, and replaces the default "*" tap point with a "?" or "!" tap point upon detecting a corresponding API invocation.

In a multi-tasking environment, if there is more than one instance of a task running when its default Running Task tap point is tapped, then in preferred embodiments a predetermined selected one is surfaced, as has been described above. Rather than surfacing a selected event, a marquee-style presentation of all such executing instances for this task may be provided, where the user may then select an instance for viewing. For example, the user may hold the pen on this tap point while the marquee shows one instance after another, where he then lifts his pen to signal that the marquee should stop, and the last-viewed instance is brought to the top of the task display stack (and is thus available for the user to see).

Referring again to FIG. 3, element 320 shows that the tap point symbols supported in the Stop Task column 140 of advanced embodiments are preferably (1) the default symbol, which again may be a dot, and which may be tapped to indicate that a selected currently-running instance of the task in this row is to be stopped, (2) the "!" symbol, which has been described above, and (3) the "C" symbol, which has also been described above.

Finally, element 330 of FIG. 3 shows the default tap point symbol "*" for closing all instances of an executing task.

Optionally, a pervasive computing device using the application launcher of the present invention may operate in a networking environment. A simple example of such a network is represented in FIG. 4, wherein example pervasive computing devices are shown as a Web-enabled cell phone 405 and PDA 410, connected using wireless network connections to a cellular tower 420 and then by land connections to a server 425 (which may be an edge server, proxy, firewall, or other networking device). Server 425 then connects to one or more other server-side devices 430 and 435, which may be Web application servers, legacy host application or database servers, and so forth.

FIGS. 5–12 provide flowcharts depicting logic that may be used in implementing preferred embodiments of the present invention. These flowcharts illustrate support for an advanced embodiment in which all tap point symbols and life cycle points discussed herein are supported. It will be obvious to one of ordinary skill in the art how the logic in the flowcharts which have been provided may be modified to provide a basic embodiment which supports only the default tap point symbol. (For example, all logic pertaining to symbols other than the default symbol may simply be omitted.) It will also be obvious how this logic may be modified to support a subset of the symbols, as well as additional or different symbols and life cycle points that may be supported in a particular implementation. Support for the basic embodiment (or for some subset of the features illustrated in FIGS. 5–12 for an advanced embodiment) may be particularly desirable on pervasive computing devices which have limited processing and/or storage capacity.

11

The logic of FIG. 5 is invoked when the user taps on the display screen of the pervasive computing device (and the operating system therefore detects an event such as "stylus touchdown"), as shown at 505. At Block 510, a variable "STP" (stylus touchdown point) is set to the (X, Y) coordinate location of the tap. Block 515 tests whether a task view pane is currently displayed. If not (i.e. the application launcher pane is displayed), then control transfers to FIG. 6, as shown at 525. Upon returning from this invocation, the processing of FIG. 5 will end, as shown at 555. Otherwise (i.e. the task view pane is displayed), processing continues to Block 520 which checks to see if the STP is within the "return to launcher" indicator (such as the left-facing arrow 210 of FIG. 2A). If so, then Block 530 causes the application launcher view to be redisplayed, and processing continues at Block 535. Otherwise, Block 550 passes the tap event to the executing task for application-specific processing, and the processing of FIG. 5 ends. (As an example of this application-specific processing, if the user taps on element 245 of FIG. 2B, the application may then display the error message 260 shown in FIG. 2C.)

When processing reaches Block 535, a variable "TT" (terminate task) is evaluated to see if it references a task to be stopped. This variable is initially set to null, and may be set to reference a task by the processing in Block 1055 of FIG. 10 (as described below). If the variable currently has a null value, then the logic of FIG. 5 merely ends (see 555). Otherwise, the referenced task is terminated (Block 540) and the TT variable is reset to null (Block 545). The processing of FIG. 5 then ends for this tap event.

The logic of FIG. 6 is invoked from Block 525 of FIG. 5 to process a tap received while the application launcher view is displayed. At Block 605, a test is made to see if the STP indicates that the tap occurred over a tap point. If it did, then processing continues to FIG. 7, as indicated at 610. Otherwise, processing continues to Block 615, which tests to see if the tap may have been over another defined point. FIGS. 13A and 13B illustrate examples of such other defined points, which may optionally be supported by an implementation of the present invention. It may happen that a particular user has more tasks than can be displayed on a single view, in which case arrows (such as those illustrated at 1310 and 1320 of view 1300 in FIG. 13A) may be displayed. Or, a slider bar such as that shown at 1360 of FIG. 13B, which also has up and down arrows 1365 and 1370, may be used alternatively. Use of these graphical elements serves two purposes: it indicates to the user that additional information is available beyond what is displayed, and it provides a means for allowing the user to display that additional information.

Block 620 is reached when the user tapped over the "scroll up" indicator (illustrated by arrow 1310 of FIG. 13A and arrow 1365 of FIG. 13B). Block 625 then checks to see if the list of tasks being displayed is already at the top. If so, then Block 630 indicates that an audible beep (or other indicator) may be provided to inform the user. When the list was not at the top, Block 635 scrolls the task list upward (preferably, by one task at a time).

Block 640 is reached when the user tapped over the "thumb area" or slider bar indicator (illustrated at 1360 of FIG. 13B). Block 645 then calculates the relative task list position ("RTL") from the point where the slider was tapped, and Block 650 positions the task list accordingly. These techniques for slider bar manipulation are well known in the art, and will not be described in detail herein.

Block 655 is reached when the user tapped over the "scroll down" indicator (illustrated by arrow 1320 of FIG.

12

13A and arrow 1370 of FIG. 13B). Block 660 then checks to see if the list of tasks being displayed is already at the bottom. If so, then Block 665 indicates that an audible beep (or other indicator) may be provided to inform the user. When the list was not at the bottom, Block 670 scrolls the task list downward (preferably, by one task at a time).

Upon completing the processing of Block 630, 635, 650, 665, or 670, control returns to FIG. 5, as indicated as 680, after which the processing for this tap event is complete.

The logic of FIG. 7 is invoked from Block 610 of FIG. 6 to handle a tap received over a tap point on the application launcher. Block 705 sets a variable "TAPCOL" (tap column) to the column location of the STP. Block 710 then sets a variable "TC" (tap character) to the character or symbol at the point where the tap event was received. Block 715 sets a variable "TROW" (tap row) to the row in which the tap event was received, thereby reflecting the corresponding task from the task list. Block 720 sets a variable "TNAME" (task name) to the name of that task. Control then branches out from Block 725, depending on which column the tap was received over. If it was the Start Task column 120, then Block 730 receives control, and (at Block 735) the processing shown in FIG. 8 is invoked. If the tap was received over the Running Task column 130, then Block 740 receives control, and (at Block 745) the processing shown in FIG. 9 is invoked. Similarly, if the tap was received over the Stop Task column 140 or the Stop All Tasks column 150, then Blocks 750 and 755, or Blocks 760 and 765, invoke the processing shown in FIG. 10 or 12, respectively.

The logic of FIG. 8 processes tap events received over the Start Task column 120. Block 805 checks to see if the tap event was over the default tap point symbol. If so, then Block 810 launches the task from this row, and then transfers control to Block 825 where the task view pane for that task is displayed. Otherwise, when the tap was not over the default tap point, Block 815 checks to see if the tap event was over the "E" symbol. If it was, Block 820 launches the task from this row, and provides any externally-supplied data that may have been received. The task view pane for the launched task is then automatically displayed (Block 825), after which processing returns to the logic of FIG. 5 (as indicated at 835), and the processing of this tap event is complete. If the tap event was not over the default character or the "E", then thus is an invalid event and Block 830 preferably signals the user, after which processing returns to FIG. 5.

The logic of FIG. 9 processes tap events received over the Running Task column 130. Block 905 checks to see if the tap event was over the default tap point symbol. If so, then Block 910 preferably displays a modal task list attached to the task name associated with TROW. FIG. 1D illustrates this type of modal task list at element 170. After the user selects an instance from this list (Block 915), then a variable "TI" (task instance) is set to refer to that instance (Block 920). The task view pane for this instance is then moved to the top of the task display stack (Block 925), and the task view pane is displayed (Block 930).

Block 935 is reached when the tap event was not over the default tap point symbol, and checks to see if it was the "?" or "!" symbol instead. If not, then an indication is preferably provided to inform the user (Block 955). Otherwise, Block 940 sets TI to the task instance requiring attention. (The manner of determining the appropriate instance has been discussed above.) Block 945 then moves the task view pane for this instance to the top of the task display stack, after which the task view pane is displayed (Block 950).

After the processing of Block 930, 950, or 955, control returns to FIG. 5 (as shown at 960), where the processing of this tap event is then complete.

13

The logic of FIG. 10 processes tap events received over the Stop Task column 140. Block 1005 checks to see what tap point the tap event was over. If it was the default tap point symbol, Block 1010 receives control, and (in Block 1015) invokes the processing of FIG. 11.

If the tap event was received over the "C" character, then Block 1020 receives control. To process a tap event over the "C" character, Block 1025 sets TI to the task by this name (i.e. having the name TNAME saved in Block 720 of FIG. 7) which generated the API invocation causing the "C" to be displayed. At Block 1030, the task instance indicated by TI is terminated, and preferably an indicator such as an audible beep is emitted (Block 1035) to notify the user. Control then returns to FIG. 5, as indicated at 1065, after which the processing for this tap event is complete.

Block 1040 is reached when the tap event was over a "!" symbol. Block 1045 sets TI to reference the task of this name that ended in the abnormal condition which triggered the display (e.g. via an API invocation) of the "!", and Block 1050 moves this task to the top of the task display stack. Block 1055 sets variable TT (discussed at Block 535 of FIG. 5) to refer to this task, and Block 1060 displays that task view pane. Processing then returns to FIG. 5 (see 1065).

The logic in FIG. 11 processes a tap event received over the default symbol in the Stop Task column 140. Block 1105 sets TI to reflect the task of this name which is closest to the top of the task display stack (i.e. the most recently viewed instance). (Other techniques for determining which instance the tap event applies to have been described above, and may be easily reflected by appropriate modification of FIG. 11.) The task view pane for this task is then moved to the top of the task display stack (Block 1110) and displayed (Block 1115). Preferably, a modal graphical element (e.g. an "OK" button or similar graphic) is displayed (Block 1120), enabling the user to confirm whether the task should be terminated or not. Block 1125 checks to see what was entered. If the user clicked on the "OK" button, then the task is terminated (Block 1130). Otherwise, an indication such as an audible beep may be emitted (Block 1135). In either case, the application launcher view is re-displayed (Block 1140), after which control returns to FIG. 5 (as indicated at 1145) and the processing of the tap event is complete.

The logic in FIG. 12 processes tap events received over the Stop All Tasks column 150. At Block 1205, TI is set to reflect the task currently at the top of the task display stack. Block 1215 then checks to see if this task's name matches the value previously stored in TNAME (set in Block 720 of FIG. 7). If so, then this is one of the tasks to be stopped, and Block 1220 terminates this instance. If not (and also after Block 1220), control reaches Block 1225 which checks to see if there are any more tasks in the task display stack. If not, then control returns to FIG. 5 (as indicated at 1230), after which processing for this tap event is complete. If there are more tasks in the task display stack, then the test in Block 1225 has a positive result and control reaches Block 1210 which sets TI to refer to the next task in the stack. The logic at Block 1215 then iterates again, thereby locating each task to be stopped.

As has been demonstrated, the present invention provides an improved user interface and improved techniques for interacting with users of pervasive computing devices. Multiple functions per task are supported, and status information is provided, all from the application launcher view.

As will be appreciated by one of skill in the art, embodiments of the present invention may be provided as methods, systems, or computer program products. Accordingly, the present invention may take the form of an entirely hardware

14

embodiment, an entirely software embodiment or an embodiment combining software and hardware aspects. Furthermore, the present invention may take the form of a computer program product which is embodied on one or more computer-usable storage media (including, but not limited to, disk storage, CD-ROM, optical storage, and so forth) having computer-usable program code embodied therein.

The present invention has been described with reference to flowchart illustrations and/or flow diagrams of methods, apparatus (systems) and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or flow diagrams, and combinations of blocks in the flowchart illustrations and/or flows in the flow diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, embedded processor or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions specified in the flowchart and/or flow diagram block(s) or flow(s).

These computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including instruction means which implement the function specified in the flowchart and/or flow diagram block(s) or flow(s).

The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide steps for implementing the functions specified in the flowchart and/or flow diagram block(s) or flow(s). Furthermore, the instructions may be executed by more than one computer or data processing apparatus.

While the preferred embodiments of the present invention have been described, additional variations and modifications in those embodiments may occur to those skilled in the art once they learn of the basic inventive concepts. Therefore, it is intended that the appended claims shall be construed to include both the preferred embodiments and all such variations and modifications as fall within the spirit and scope of the invention.

We claim:

1. A user interface displayed on a pervasive computing device, the user interface comprising:
 - a graphical representation identifying each of a collection of one or more tasks which are currently executing or executable on the pervasive computing device; and
 - an execution status display area corresponding to the graphical representation of the collection, wherein the execution status display area indicates execution status information for each of the one or more tasks, including whether the task is currently executing or is executable, and also provides graphical symbol indicating, for each of the tasks in the collection, each of one or more available functions that can be performed on that task, whereby a user of the pervasive computing device can perform any of the available functions on the tasks by

15

activating its graphical symbol, and wherein an available function for executable ones of the tasks is to cause that task to begin executing and an available function for currently-executing ones of the tasks is to cause that task to stop executing.

2. An improved user interface for a pervasive computing device, the improved user interface comprising a multi-functional application launcher view and a plurality of task views, wherein a user of the pervasive computing device selectively navigates between the multi-functional application launcher view and selected ones of the task views, and wherein:

the multi-functional application launcher view comprises:

a displayed list of tasks available for execution by the pervasive computing device;

a displayed grid corresponding to the displayed list of tasks, wherein the displayed grid comprises a plurality of columns and a row for each task in the displayed task list, each of the columns corresponding to life cycle points for the displayed list of tasks; and

displayed entries in the grid at selected intersections of the columns and the rows, wherein the displayed entries provide execution status information about particular life cycle points of the task associated with that row or graphical depictions that may be activated to perform actions for particular life cycle points of the task associated with that row; and

each of the task views provides information pertaining to that one of the tasks that is associated with the task view.

3. The improved user interface according to claim 2, wherein selected ones of the graphical depictions enable performing an action that launches execution of the task associated with that row.

4. The improved user interface according to claim 2, wherein at least one of the displayed enables performing a plurality of actions relative to a selected one of the tasks in the displayed list.

5. The improved user interface according to claim 2, wherein the actions include (1) starting execution of a selected task and (2) surfacing the task view for a selected task.

6. The improved user interface according to claim 5, wherein the actions also include stopping execution of a selected task.

7. An improved user interface for a pervasive computing device, the improved user interface comprising a multi-functional application launcher view and a plurality of task views, wherein a user of the pervasive computing device selectively navigates between the multi-functional application launcher view and selected ones of the task views, and wherein the multi-functional application launcher view comprises:

a displayed list of tasks available for execution by the pervasive computing device; and

a displayed grid corresponding to the displayed list of tasks, wherein the displayed grid comprises a plurality of columns and a row for each task in the displayed task list and entries at intersections of selected ones of the columns and the rows, wherein the entries provide status information for the tasks in the displayed list, wherein the status information includes (1) which tasks are executing, (2) which tasks are requesting input from the user, and (3) which tasks have information to display to the user.

8. The improved user interface according to claim 7, wherein the status information also includes a notification that a particular task should be started by the user.

16

9. The improved user interface according to claim 8, wherein the notification is received because of an event which occurred in another task.

10. The improved user interface according to claim 8, wherein the notification is received because a message arrived for the user, and wherein the message indicates that the particular task should be started.

11. The improved user interface according to claim 10, wherein the message includes input values to be used by the particular task when it is started.

12. The improved user interface according to claim 11, wherein the input values are encoded within a structured markup language document.

13. The improved user interface according to claim 7, wherein the status information also includes which tasks have completed normally.

14. The improved user interface according to claim 7, wherein the status information also includes which tasks have completed abnormally, and wherein the user may see the task view of a selected one of the tasks that have completed abnormally by activating the entry providing the status information of the selected task.

15. The improved user interface according to claim 4, wherein the plurality of actions comprises stopping execution of multiple instances of the selected one of the tasks.

16. The improved user interface according to claim 2, wherein the user activates a selected one of the graphical depictions, thereby causing an action to be performed by tapping the selected one with a stylus or pen input device.

17. The improved user interface according to claim 5, wherein the actions also include stopping execution of a selected group of tasks.

18. The improved user interface according to claim 17, wherein the selected group comprises multiple instances of a particular task from the displayed list.

19. The improved user interface according to claim 5, further comprising a graphical selection area on the surfaced task view for the selected task that may be activated by the user to return to the multi-function application launcher view.

20. A method of providing an improved user interface for a pervasive computing device, comprising steps of:

displaying a multi-functional application launcher view, comprising steps of:

displaying a list of tasks available for execution by the pervasive computing device;

displaying a grid corresponding to the displayed list of tasks, wherein the displayed grid comprises plurality of columns and a row for each task in the displayed task list, each of the columns corresponding to life cycle points of the displayed list of tasks; and

displaying entries in the grid at selected intersections of the columns and the rows, wherein the displayed entries provide status information about particular life cycle points of the task associated with that row or graphical depictions of available actions for particular life cycle points of the task associated with that row; and

enabling a user of the pervasive computing device to selectively navigate between the multi-functional application launcher view and selected ones of a plurality of task views, wherein each task view provides information pertaining to one of the tasks in the displayed list.

21. The method according to claim 20, further comprising the step of enabling the user to modify the life cycle points.

22. The method according to claim 20, wherein the user activates one of the available actions by activating its graphical depiction in the grid.

17

23. The method according to claim 20, further comprising the step of revising a selected displayed entry when the task associated with that row has different status information.

24. The method according to claim 20, further comprising the step of revising a selected displayed entry when the task associated with that row has a different available action. 5

25. The method according to claim 24, wherein the step of revising further comprises the step of receiving an application programming interface invocation indicating the different available action. 10

26. The method according to claim 20, further comprising the step of displaying a graphical indication on the multi-functional application launcher view when the displayed list of tasks exceeds a display capacity of the pervasive computing device. 15

27. The method according to claim 20, wherein the available actions include (1) starting execution of a selected task and (2) surfacing the task view for a selected task.

28. The method according to claim 27, wherein the actions also include stopping execution of a selected task. 20

29. The method according to claim 27, wherein a plurality of instances of a selected task are executing, and further comprising the step of surfacing the task view of a particular one of the plurality when the user activates the graphical depiction for surfacing the task view of the selected task. 25

30. The method according to claim 29, wherein the particular one is the instance which was most recently viewed by the user.

31. The method according to claim 29, wherein the activated graphical depiction indicates that input is requested from the user, and wherein the particular one is the instance which is requesting the input. 30

32. The method according to claim 29, wherein the activated graphical depiction indicates that information is available for presenting to the user, and wherein the particular one is the instance which has the available information. 35

33. The method according to claim 29, wherein the particular one is selected by the user.

34. The method according to claim 33, wherein: 40
the activated graphical depiction indicates presence of the plurality;

a selectable representation of the plurality is presented to the user in response to the activation; and

the particular one is the instance which is selected by the user from the selectable representation. 45

35. The method according to claim 20, wherein the status information includes (1) which tasks are executing, (2) which tasks are requesting input from the user, and (3) which tasks have information to display to the user. 50

36. A system for providing an improved user interface for a pervasive computing device, comprising:

18

means for displaying a multi-functional application launcher view, comprising:

means for displaying a list of tasks available to the user for execution by the pervasive computing device;

means for displaying a grid corresponding to the displayed list of tasks, wherein the displayed grid comprises a plurality of columns and a row for each task in the displayed task list, each of the columns corresponding to life cycle points for the displayed list of tasks;

means for displaying entries in the grid at selected intersections of the columns and the rows, wherein the entries provide (1) status information about particular life cycle points of the task associated with that row, and (2) graphical depictions that may be activated to perform actions for particular life cycle points of the task associated with that row; and

means for enabling a user of the pervasive computing device to selectively navigate between the multi-functional application launcher view and selected ones of a plurality of task views.

37. A computer program product for providing an improved user interface for a pervasive computing device, the computer program product embodied on one or more computer-readable media and comprising:

computer-readable program code means for displaying a multi-functional application launcher view, comprising:

computer-readable program code means for displaying a list of tasks available to the user for execution by the pervasive computing device;

computer-readable program code means for displaying a grid corresponding to the displayed list of tasks, wherein the displayed grid comprises a plurality of columns and a row for each task in the displayed task list, each of the columns corresponding to life cycle points for the displayed list of tasks; and

computer-readable program code means for displaying entries in the grid at selected intersections of the columns and the rows, wherein the entries provide (1) status information about particular life cycle points of the task associated with that row, and (2) graphical depictions that may be activated to perform actions for particular life cycle points of the task associated with that row.

38. The computer program product according to claim 37, further comprising computer-readable program code means for enabling a user of the pervasive computing device to selectively navigate between the multi-functional application launcher view and selected ones of a plurality of task views.

* * * * *



US006750830B1

(12) **United States Patent**
Teshima et al.

(10) Patent No.: **US 6,750,830 B1**
(45) Date of Patent: **Jun. 15, 2004**

(54) **IMAGE COMMUNICATION SYSTEM**

(75) Inventors: **Atsushi Teshima, Asaka (JP); Norihisa Haneda, Asaka (JP)**

(73) Assignee: **Fuji Photo Film Co., Ltd., Minami-Ashigara (JP)**

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 31 days.

(21) Appl. No.: **09/616,673**

(22) Filed: **Jul. 14, 2000**

(30) **Foreign Application Priority Data**

Jul. 15, 1999 (JP) 11-201591

(51) Int. Cl.⁷ **G09G 5/00**

(52) U.S. Cl. **345/1.2; 345/1.3**

(58) Field of Search **345/145, 158, 345/1.3, 156, 183, 864; 358/1.15, 1.14, 434, 437, 442; 710/17, 19; 700/83**

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,293,484 A * 3/1994 Dabbs, III et al. 395/164

5,689,642 A * 11/1997 Harkins et al. 395/200.04
5,726,768 A * 3/1998 Ishikawa et al. 358/442
6,049,316 A * 4/2000 Nolan et al. 345/698
6,199,099 B1 * 3/2001 Gershman et al. 709/203
6,253,326 B1 * 6/2001 Lincke et al. 713/201
6,262,805 B1 * 7/2001 Ishikawa et al. 358/1.15
6,400,996 B1 * 6/2002 Hoffberg et al. 700/83

* cited by examiner

Primary Examiner—Amare Mengistu

Assistant Examiner—Prabodh Dharia

(74) *Attorney, Agent, or Firm*—McGinn & Gibb, PLLC

(57)

ABSTRACT

In a user device which permits access to a server, an image suitable for the user device is displayed. Various user devices such as a portable telephone set, a personal digital assistant, a notebook personal computer, and a personal computer permit access to the server. The server stores data representing an image suitable for each of the devices. The data representing the image suitable for the user device which has accessed the server is read out of the server. Data representing the image read out is transmitted to the user device from the server. In the user device, the data is received, and an image suitable for the user device is displayed.

31 Claims, 53 Drawing Sheets

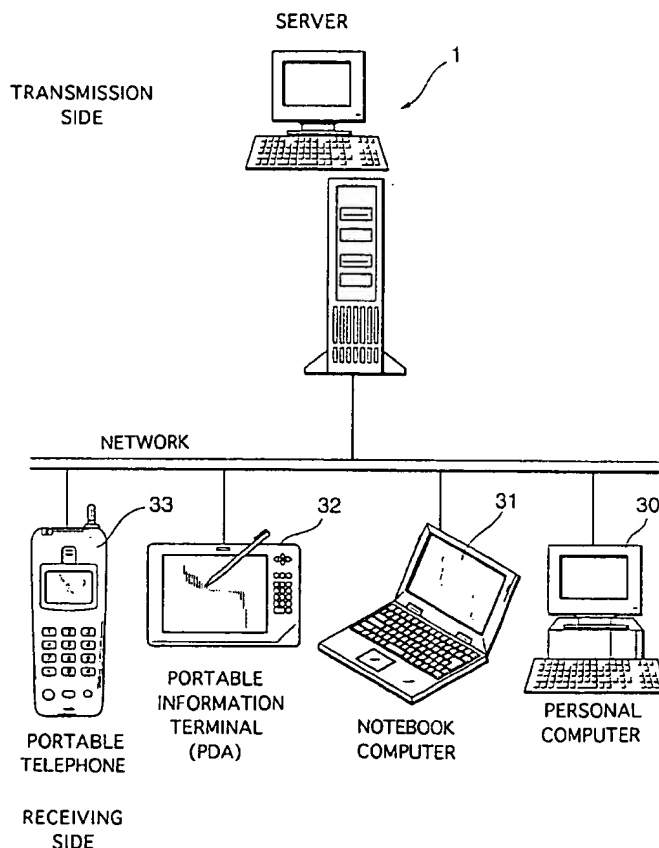
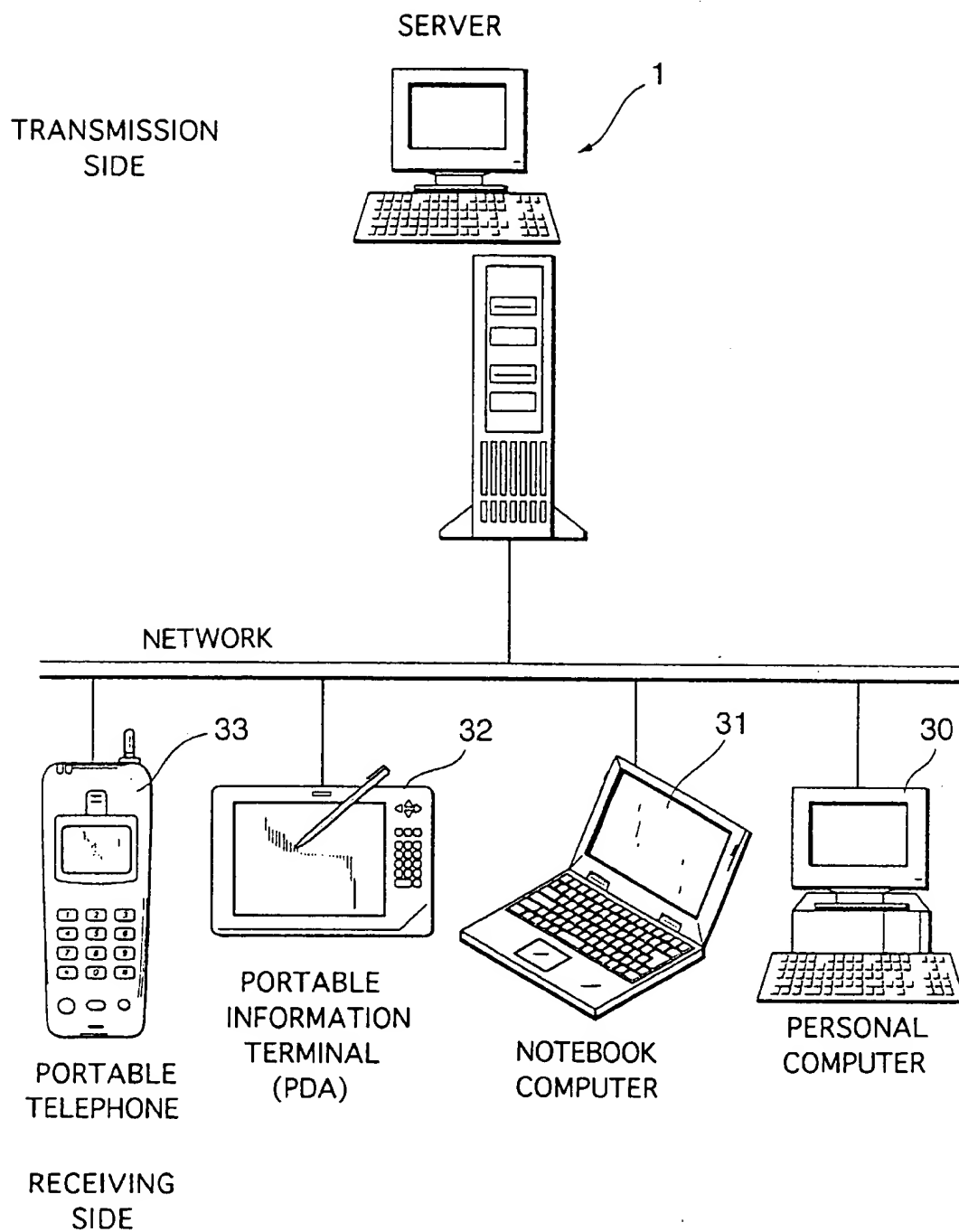


Fig. 1

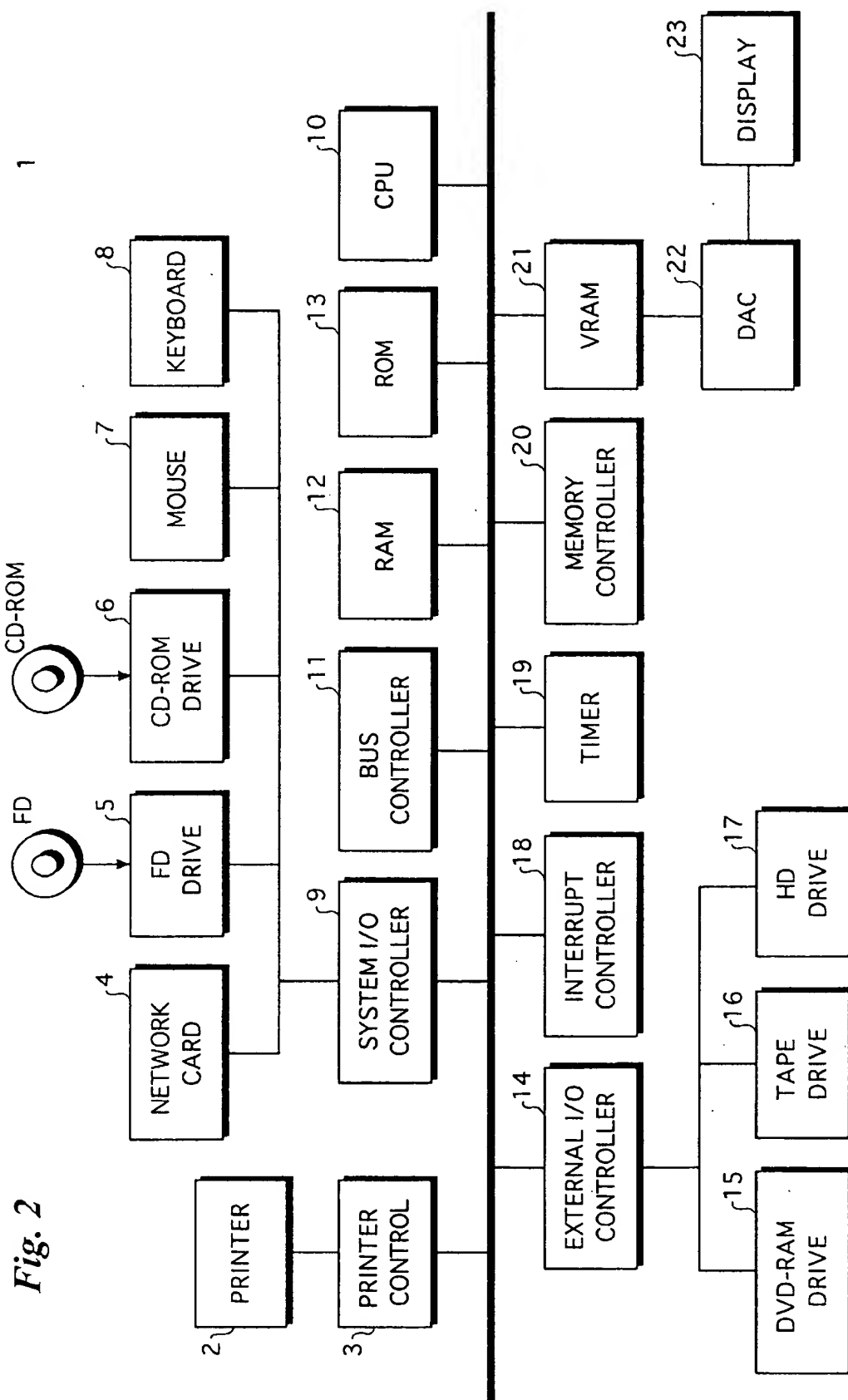


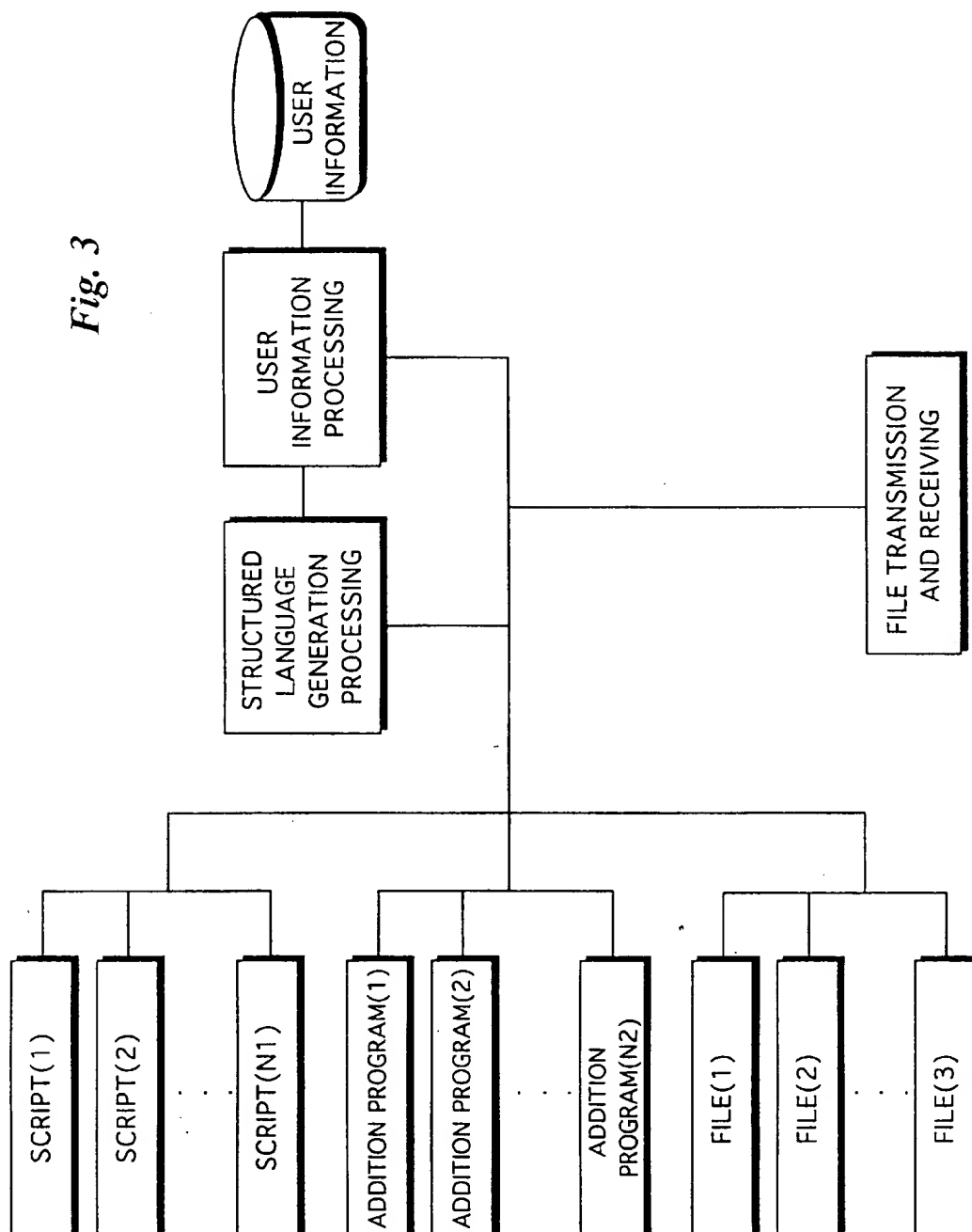
Fig. 3

Fig. 4

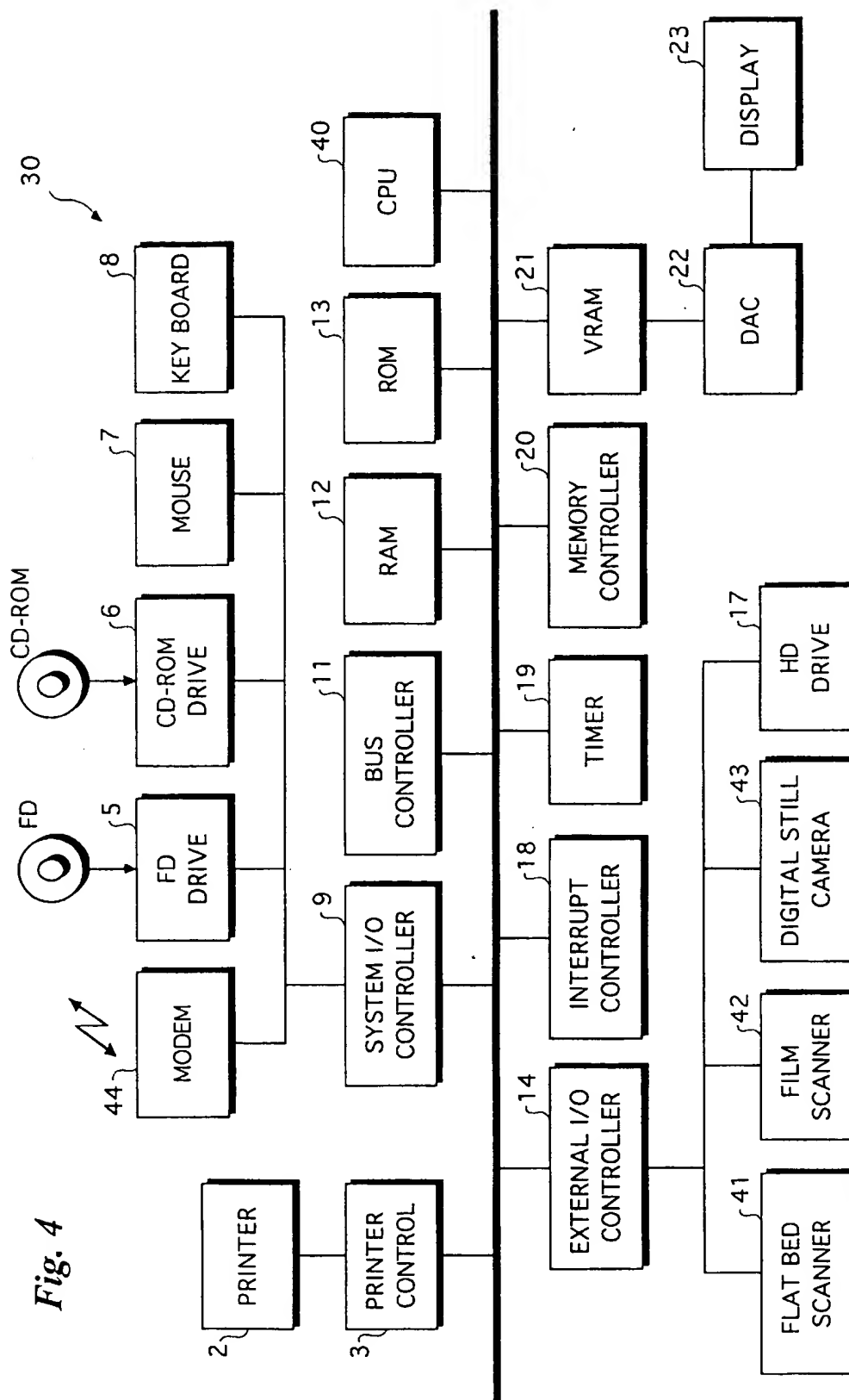


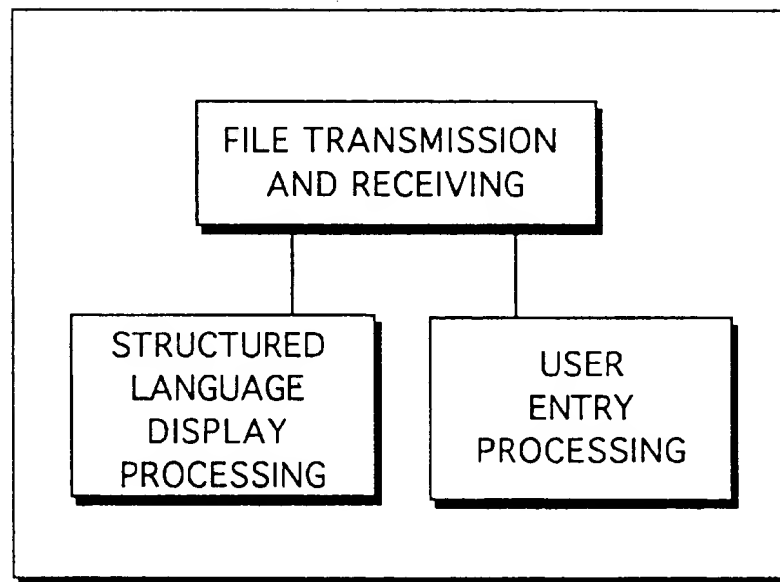
Fig. 5

Fig. 6

TRANSITION OF IMAGES

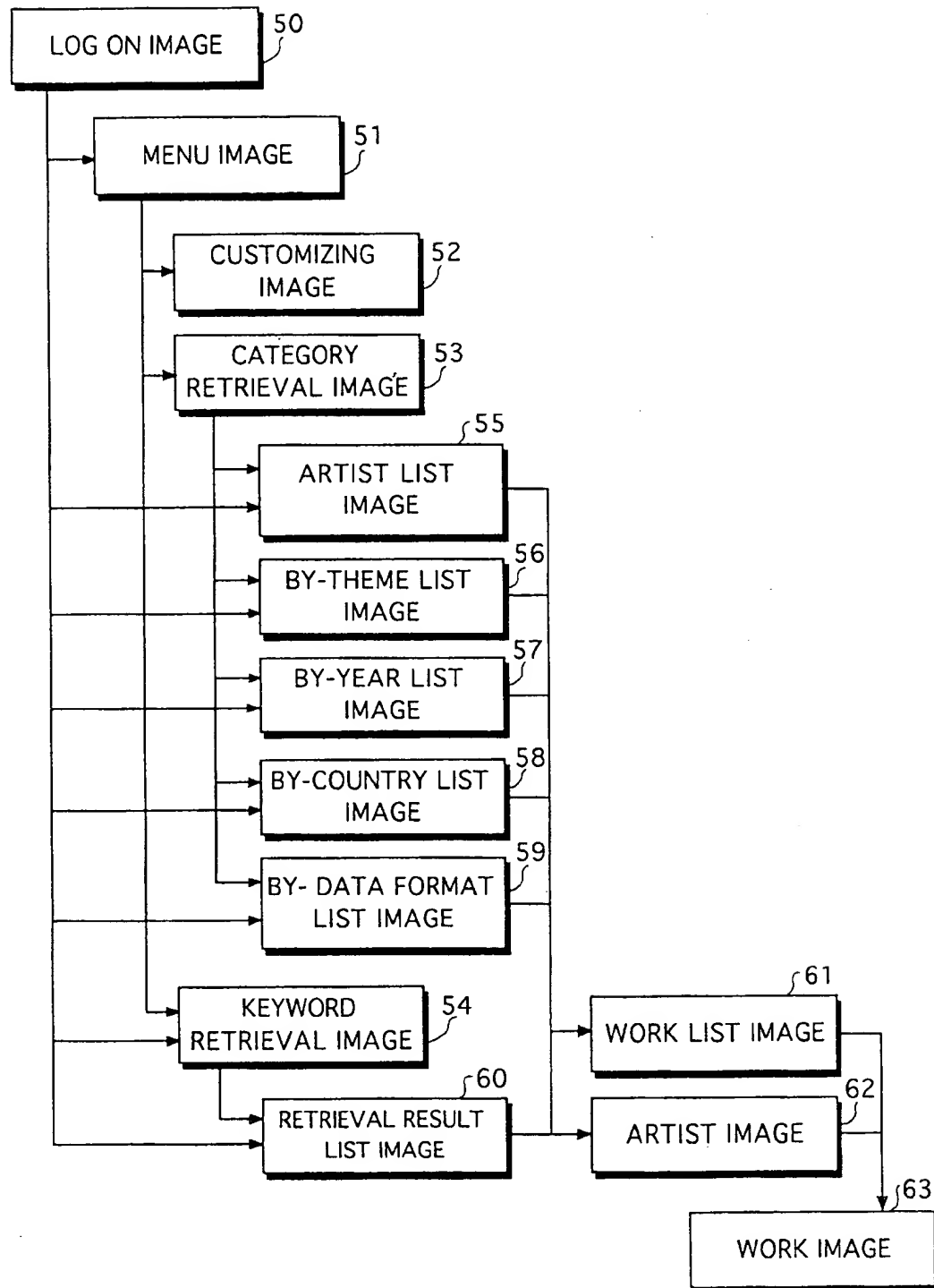


Fig. 7

USER NAME	PASSWORD	START IMAGE	RETRIEVAL CONDITIONS
A	***	CATEGORY RETRIEVAL IMAGE	FILE FORMAT = ARTIST and DATE OF BIRTH \geq 1950/1/1
B	***	CATEGORY RETRIEVAL IMAGE	FILE FORMAT = ARTIST and DATE OF BIRTH \geq 1950/1/1
C	***	CATEGORY RETRIEVAL IMAGE	FILE FORMAT = ARTIST and WORK CATEGORY = ANIMAL
D	***	CATEGORY RETRIEVAL IMAGE	FILE FORMAT = ARTIST and WORK CATEGORY = ANIMAL

MENU IMAGE	...	CATEGORY RETRIEVAL IMAGE	WORK LIST IMAGE	ARTIST IMAGE	WORK IMAGE
NO	...	H121	NO	J121	NO
NO	...	H122	NO	J121	NO
NO	...	H123	NO	J121	NO
NO	...	H124	NO	J121	NO

ARTIST IMAGE MANAGEMENT FILE

Fig. 8A

NAME OF ARTIST	NORI-HANEDA
NATIONALITY	AMERICA
DATE OF BIRTH	1959/3/6
DATE OF DEATH	-
DESCRIPTIVE TEXT FILE	Txt3021
FILE FORMAT	ARTIST
NUMBER OF IMAGE FILES	3
IMAGE FILE NAME 1	noriL.jpg
IMAGE FILE NAME 2	noriM.jpg
IMAGE FILE NAME 3	noriS.jpg

Fig. 8B

ARTIST IMAGE FILE (LARGE)



noriL.jpg 2400 × 1800

Fig. 8C

ARTIST IMAGE FILE (MEDIUM)



noriM.jpg 160 × 120

Fig. 8D

ARTIST IMAGE FILE (SMALL)

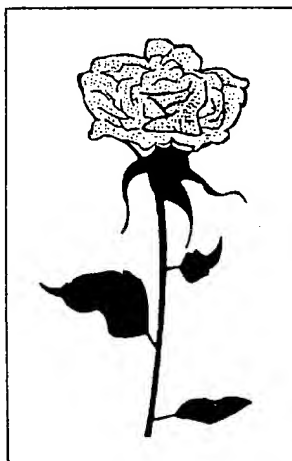
noriS.jpg
80 × 50

WORK IMAGE MANAGEMENT FILE

Fig. 9A

NAME OF ARTIST	NORI-HANEDA
TITLE	Red Rose
YEAR OF ISSUE	1974
WORK CATEGORY	FLOWER
DESCRIPTIVE TEXT FILE	Txt18416
FILE FORMAT	WORK
NUMBER OF IMAGE FILES	32
IMAGE FILE NAME 1	nori001L.jpg
IMAGE FILE NAME 2	nori001S.jpg

WORK IMAGE FILE (LARGE)

Fig. 9B

nori001L.jpg 1800 × 2560

WORK IMAGE FILE (SMALL)

Fig. 9Cnori001S.jpg
90 × 128

Fig. 10A

TEXT MANAGEMENT FILE

FILE FORMAT	TITLE
NUMBER OF TEXT FILES	1
TEXT FILE NAME	title3.txt

Fig. 10B

TEXT FILE

N _____

t i t l e 3 . t x t

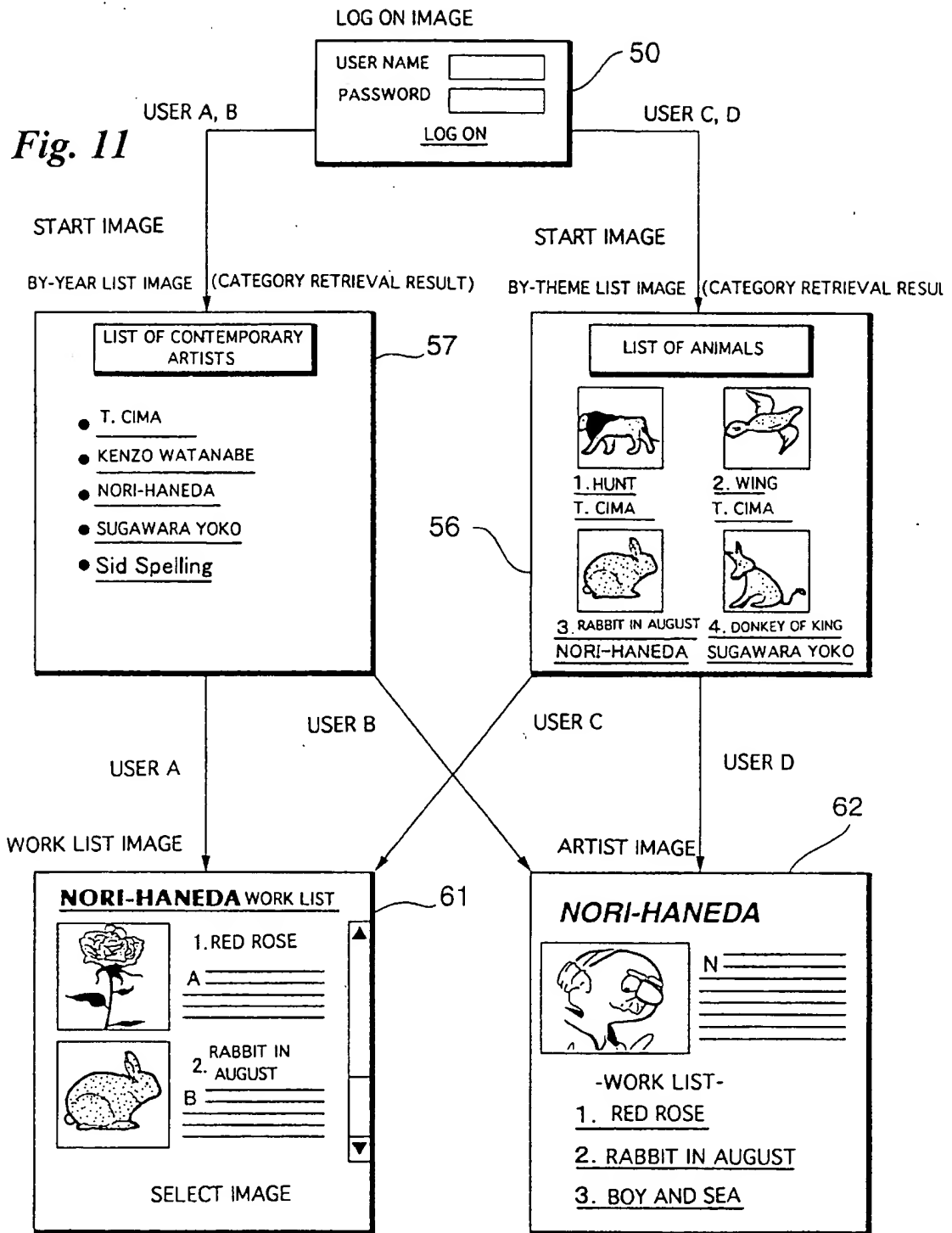


Fig. 12

H121

```
<file Txt15 function31(3)>  
<for i = 1 to ##Count>  
  <file manage(##FileID(i),ARTIST NAME)function4(16)function1  
  L (WORK LIST, FILE · FORMAT = WORK and  
  L ' ARTIST NAME = manage(##FileID(i), ARTIST NAME)' )>  
<next>
```

Fig. 13

H122

```
<file Txt15 function31(3)>  
<for i = 1 to ##Count>  
  <file manage(##FileID(i), ARTIST NAME)function4(16)function1  
  L (ARTIST, FILE · FORMAT = WORK and  
  L ' ARTIST NAME = manage (##FileID(i), ARTIST NAME)' )>  
<next>
```

Fig. 14

H123

```
<file Txt16 function31(3)>
<for i = 1 to ##Count>
  <begin column(i mod 2 + 1)>
    <FILE ##FileID(i)function11(150,150)>
    <file manage(##FileID(i),TITLE)function1
    L (WORK LIST, FILE · FORMAT = WORK and
    L ARTIST= manage(##FileID(i), ARTIST NAME)' )>
    <file manage(##FileID(i), ARTIST NAME)function
    L (WORK LIST, FILE · FORMAT = WORK and
    L ARTIST = manage(##FileID(i), ARTIST NAME)' )>
  <end column(i mode 2 + 1)
<next>
```

Fig. 15

H124

```
<file Txt16 function31(3)>
<for i = 1 to ##Count>
  <begin column(i mod 2 + 1)>
    <FILE ##FileID(i)function11(150,150)>
    <file manage(##FileID(i), TITLE)function1
    L (ARTIST, FILE · FORMAT = WORK and
    L ARTIST = manage(##FileID(i), ARTIST NAME)' )>
    <file manage(##FileID(i), ARTIST NAME)function
    L (ARTIST, FILE · FORMAT = WORK and
    L ARTIST = manage(##FileID(i), ARTIST NAME)' )>
  <end column(i mode 2 + 1)
<next>
```

Fig. 16

1121

(PORTABLE TELEPHONE)

<file manage(##FileID(1), ARTIST NAME)>

"WORK LIST"

<for #i = 1 to ##Count>

```
#i" ." <file manage(##FileID(#i), TITLE)function1(WORK,
    L' FILE FORMAT= WORK and TITLE = manage(##FileID(#i),
    LTITLE)' >
```

<next>

"SELECT NUMBER"

Fig. 17

1122

(PERSONAL DIGITAL ASSISTANT) (PORTABLE INFORMATION TERMINAL)

<file manage(##FileID(#i), ARTIST NAME)> "WORK LIST"

<for #i = 1 to ##Count>

<begin column(#i mod 3 + 1)>

```
<file ##FileID(#i) function10(2) function11(64,64) function(
    L WORK , ' FILE FORMAT= WORK and TITLE= manage(##FileID(#i)
    LTITLE)>
```

#i" ." <file manage(##FileID(i), TITLE)>

<end column(#i mod 3 + 1)>

<next>

"SELECT IMAGE"

Fig. 18

1123

(NOTEBOOK COMPUTER)

```
<file manage(##FileID(1), ARTIST NAME) function32(2) function3( 'Times' )
function4(18)>
  <"WORK LIST" function32(2) function3( 'GOTHIC' ) function4(18)>
<for #i = 1 to ##Count>
  <begin column 1>
    <file ##FileID(#i) function11(100,100) function1*WORK, 'FILE
      LFORMAT= WORK and TITLE = manage(##FileID(#i), TITLE)>
  <end column 1>
  <begin column 2>
    #i" ." <file manage(##FileID(i), TITLE)function3( 'COMIC' )>
    <file manage(##FileID(i), DESCRIPTIVE TEXT)function5(1)
      Lfunction6(100)>
  <end column 2>
  <end column(#i mod 3 + 1)>
<next>
<begin center>
  "SELECT IMAGE"
<end center>
```

Fig. 19

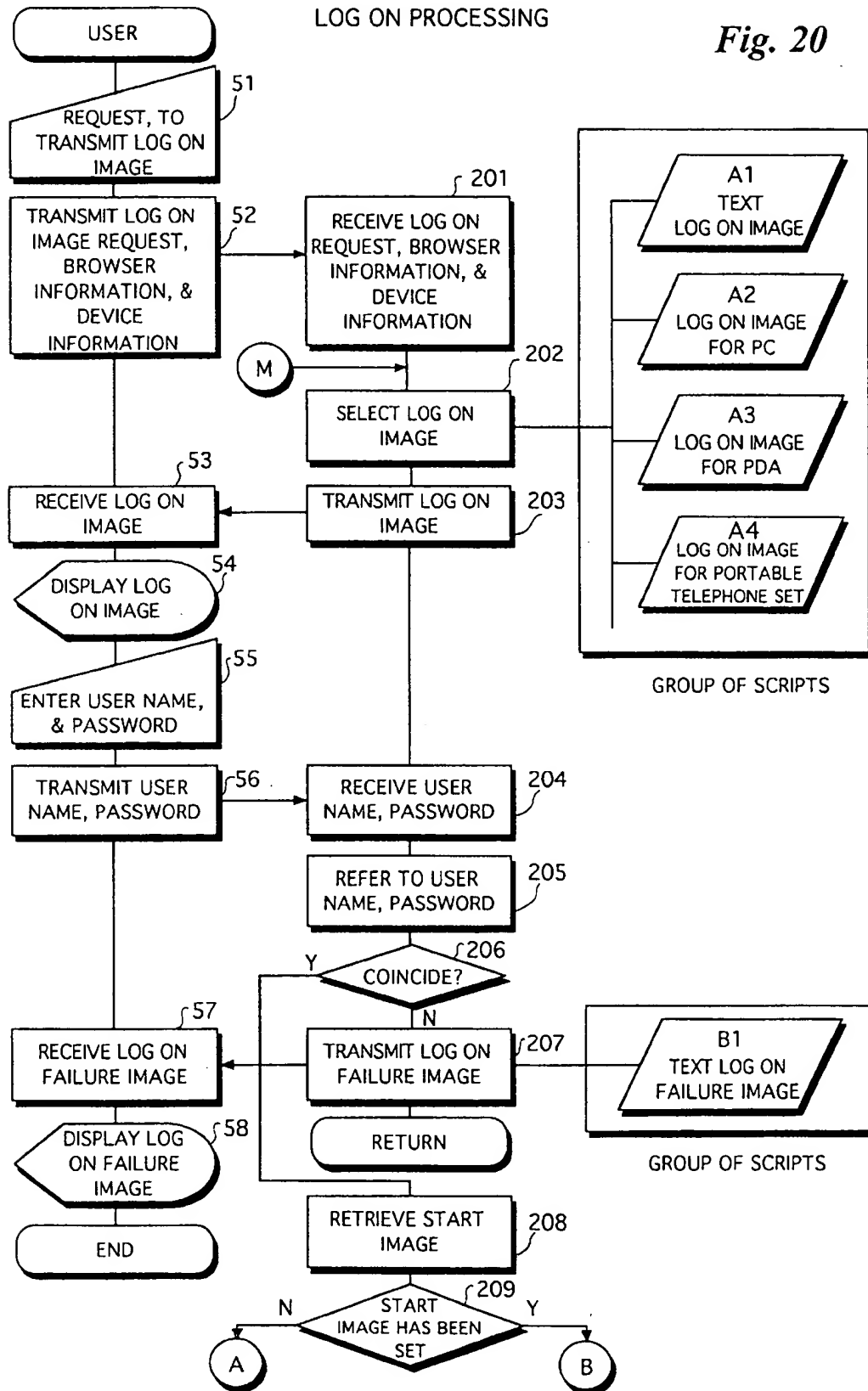
1124

(PERSONAL COMPUTER)

```
<begin column 1>
  <begin center>
    <file lmg24>
    <file lmg25 function1(MENU)>
    <file lmg25 function1(KEYWORD RETRIEVAL)>
    <file lmg25 function1(CATEGORY RETRIEVAL RESULT, 'FILE FORMAT=ARTIST' )>
  <end center>
<end column 1>
<begin column 2>
  <file manage(##FileID(#i),ARTIST NAME) function32(2) function3( 'Times' )
function4(24)>
  <"WORK LISR" function32(2) function3( 'GOTHIC' ) function4(24)>
  <for #i = 1 to ##Count>
    <begin column(#i mod 2 + 1) function31(1)>
      <begin column 1>
        <file ##FileID(#i) function11(160,160) function1(WORK,
          L 'FILE FORMAT= WORK and TITLE = manage(##FileID(#i)
          L ,TITLE)>
        <region #1>
      <end column 1>
      <begin column 2>
        # " ." <file manage(##FileID(i),TITLE) function3( 'COMIC' )>
        <file manage(##FileID(i)DESCRIPTIVE TEXT) function5(1)
          L function6(240) using region #1>
      <end column 2>
    <end column(#i mod 2 + 1)>
  <next>
  <begin center>
    " SELECT IMAGE"
  <end center>
<end column 2>
```

LOG ON PROCESSING

Fig. 20



MENU IMAGE PROCESSING

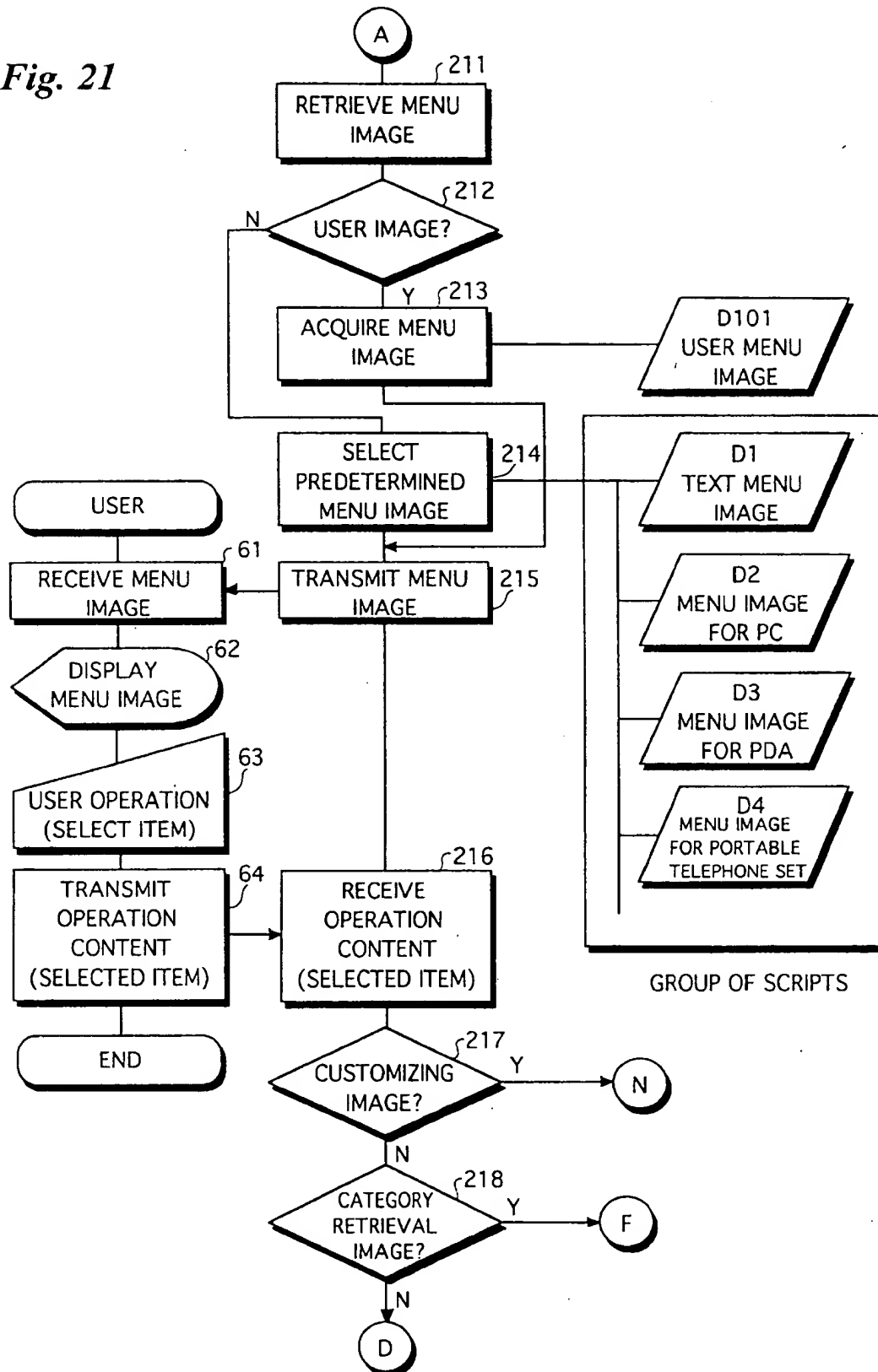
Fig. 21

Fig. 22

START IMAGE PROCESSING

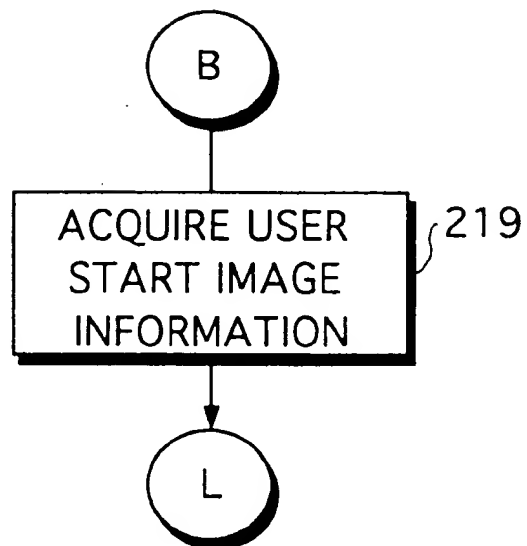
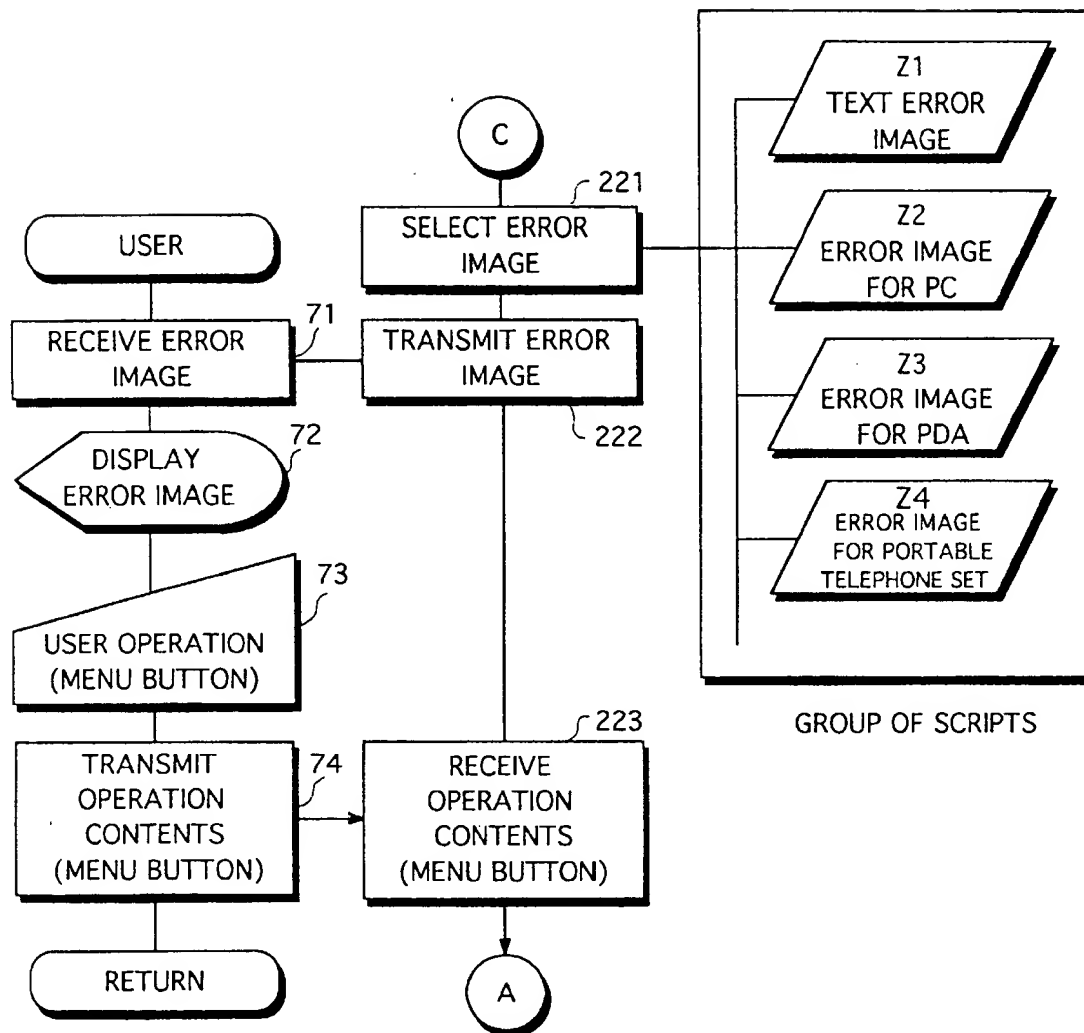


Fig. 23

ERROR IMAGE PROCESSING



KEYWORD RETRIEVAL IMAGE PROCESSING

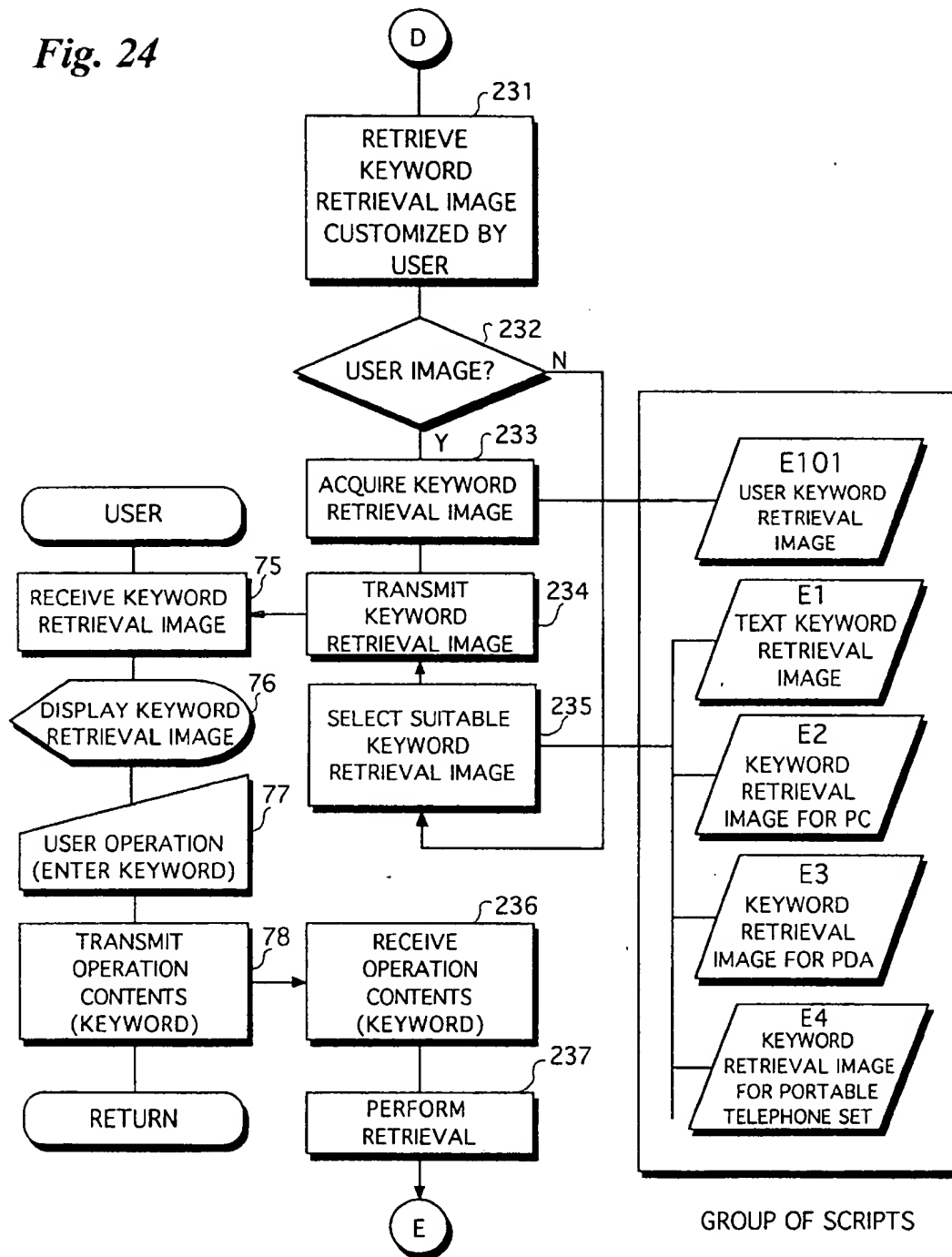
Fig. 24

Fig. 25

RETRIEVAL RESULT IMAGE PROCESSING

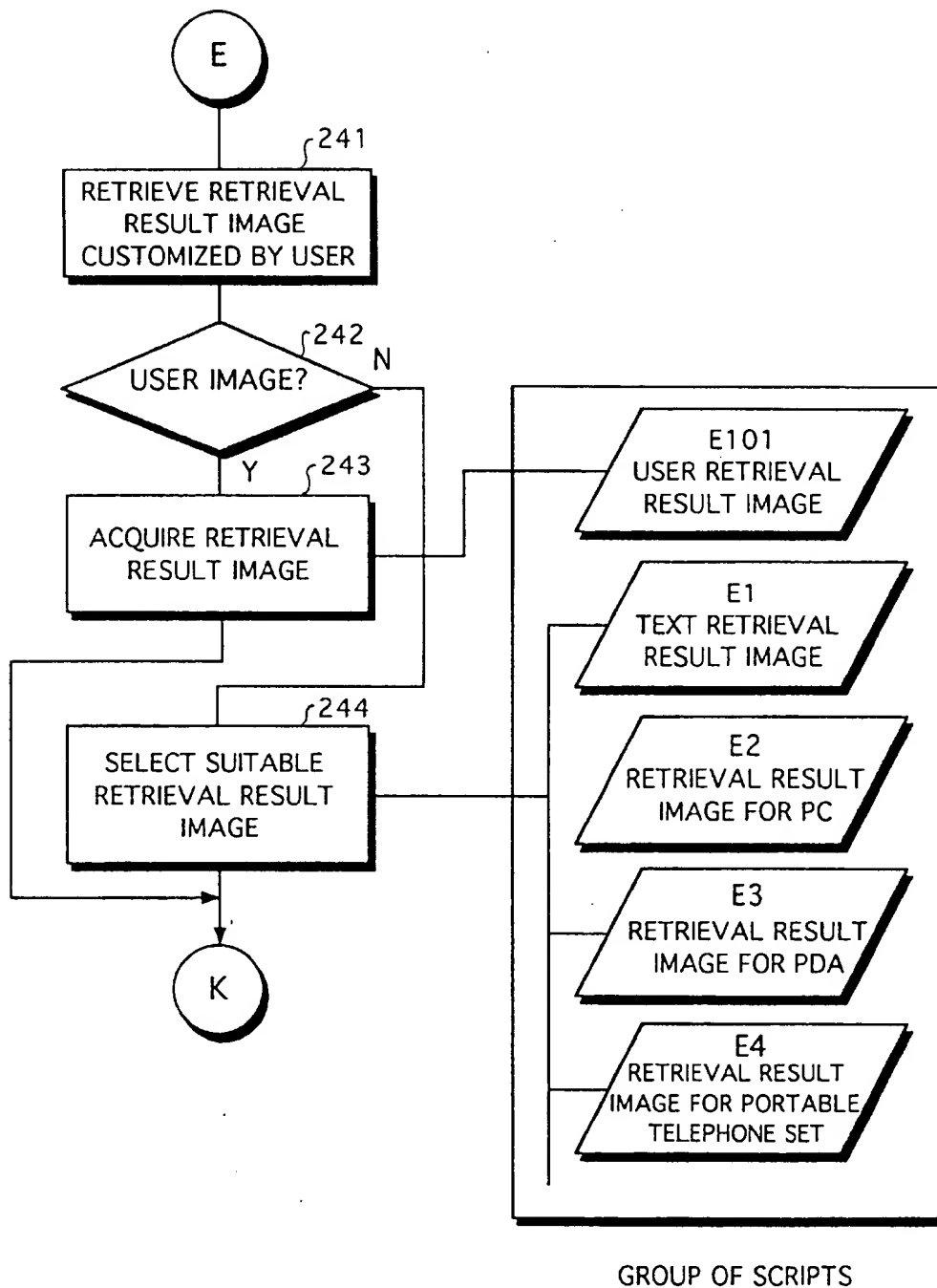


Fig. 26

CATEGORY RETRIEVAL IMAGE PROCESSING

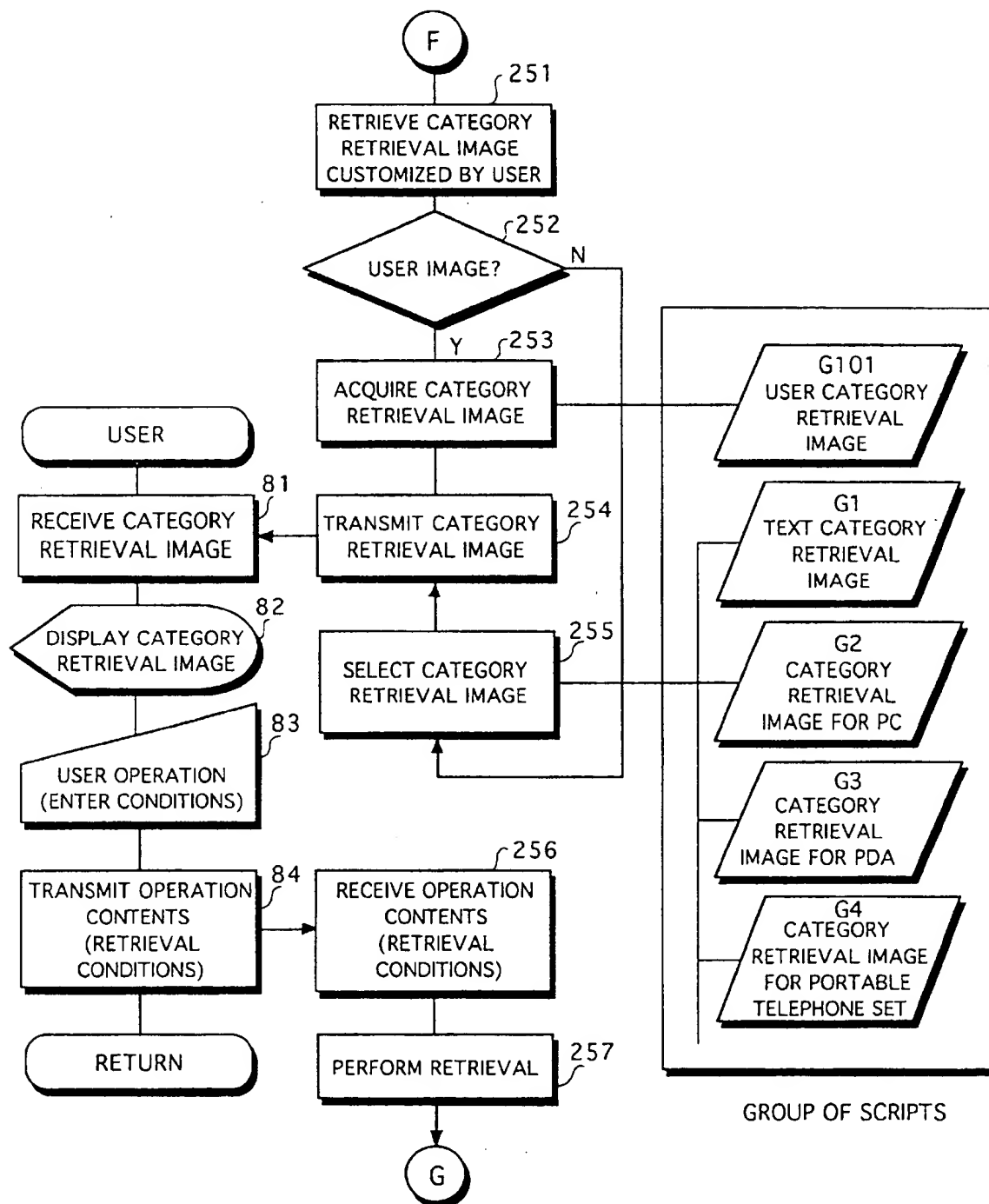


Fig. 27

CATEGORY RETRIEVAL RESULT PROCESSING

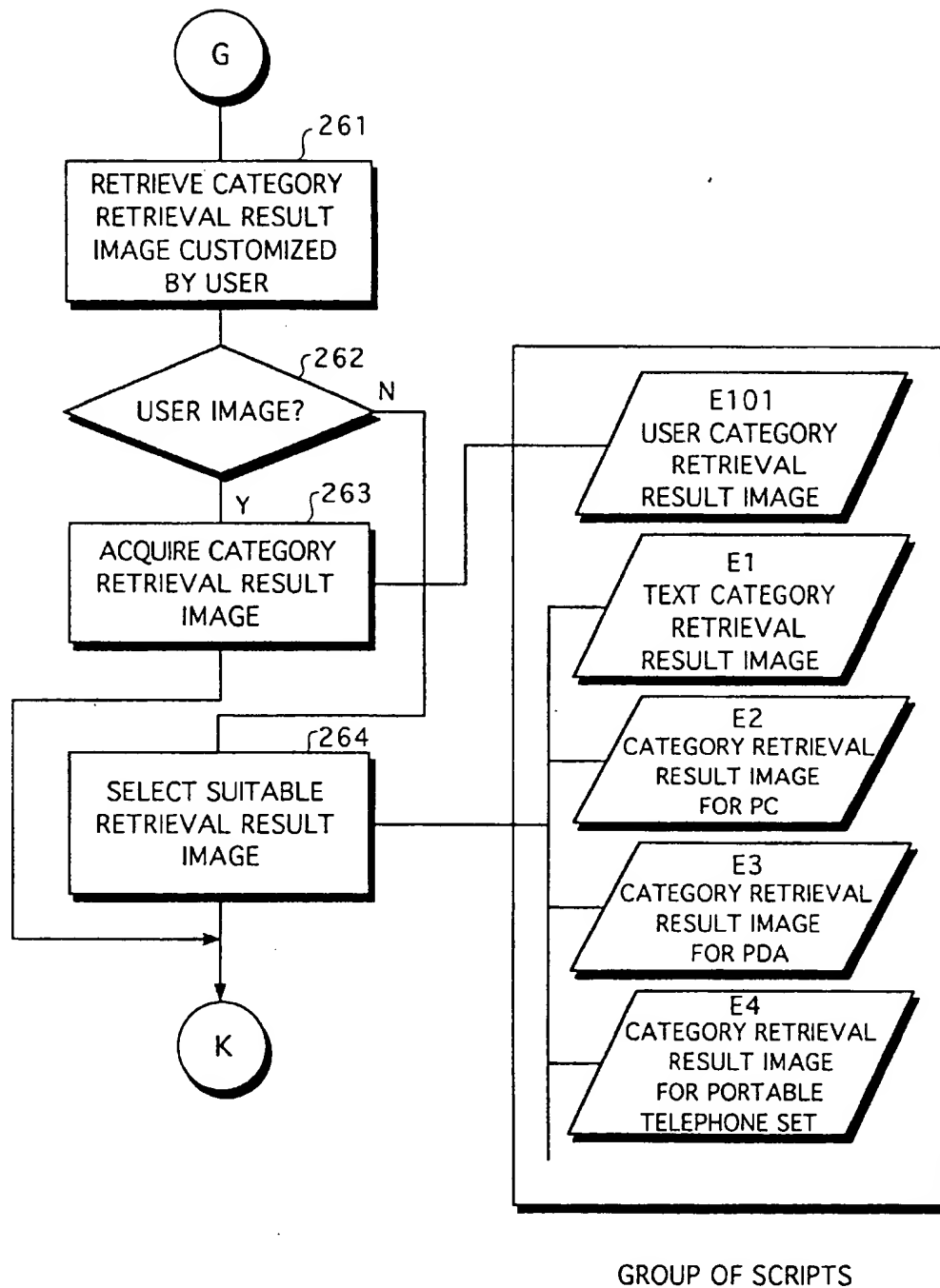


Fig. 28

WORK LIST PROCESSING

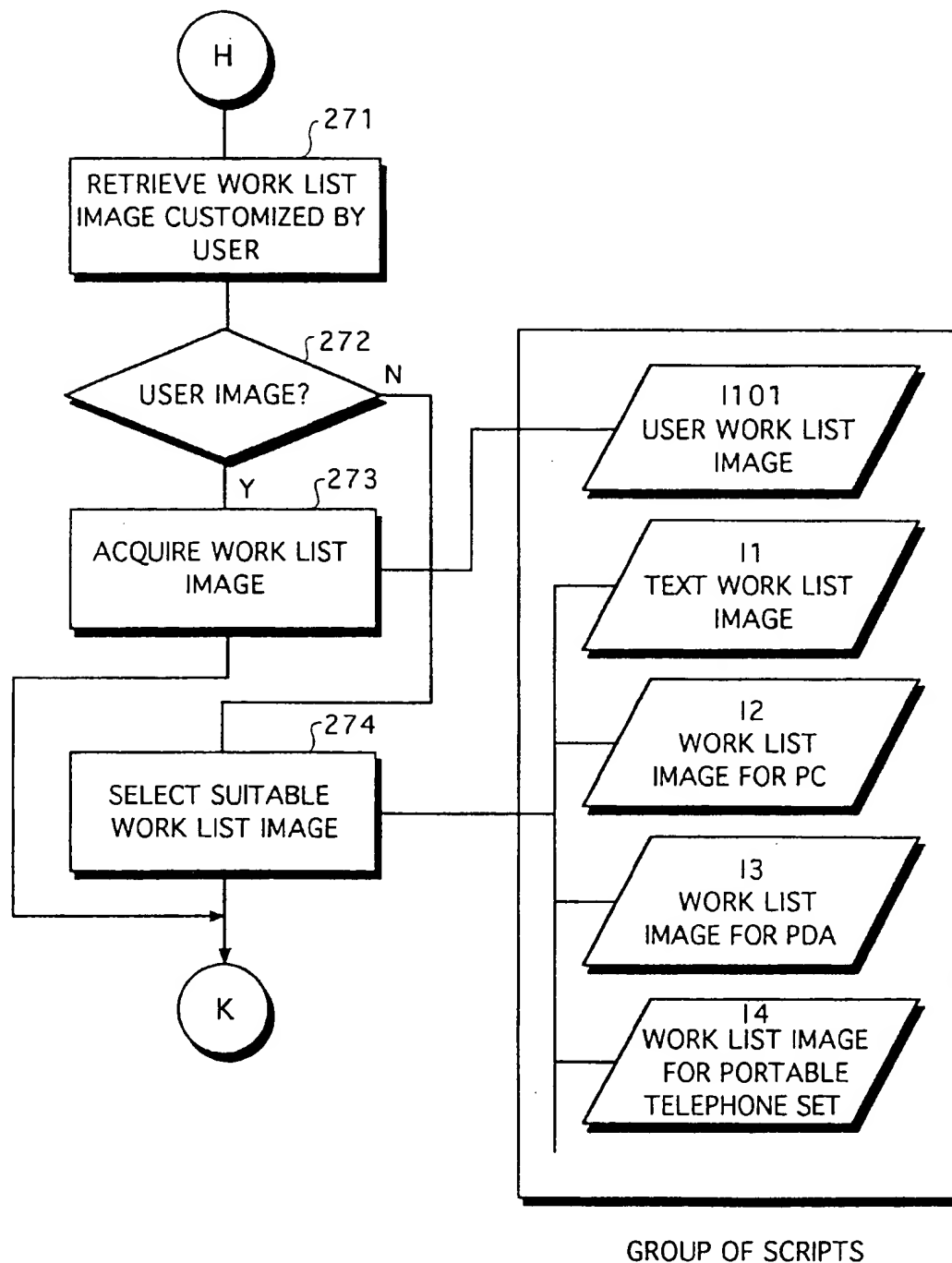


Fig. 29

ARTIST IMAGE PROCESSING

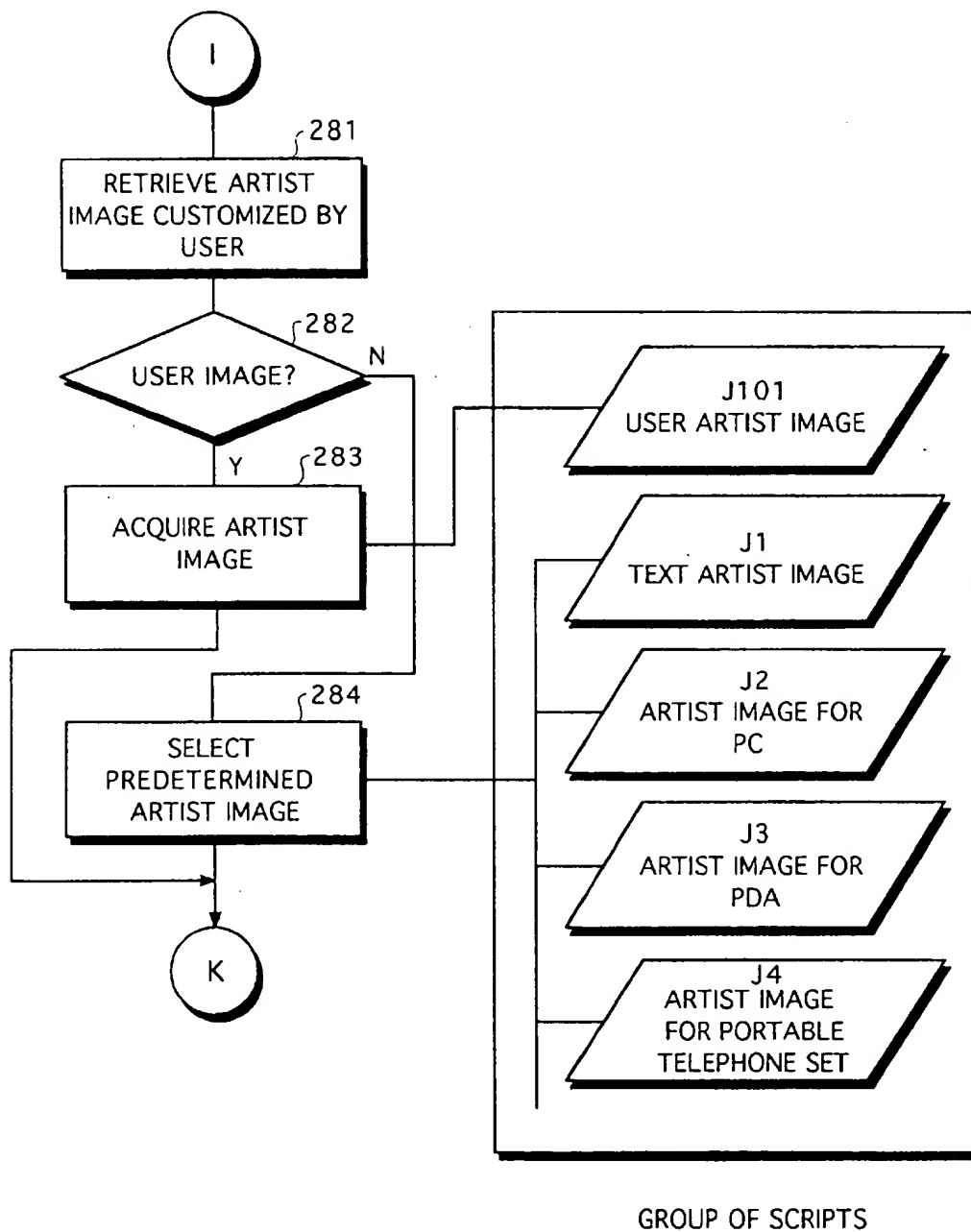


Fig. 30

WORK IMAGE PROCESSING

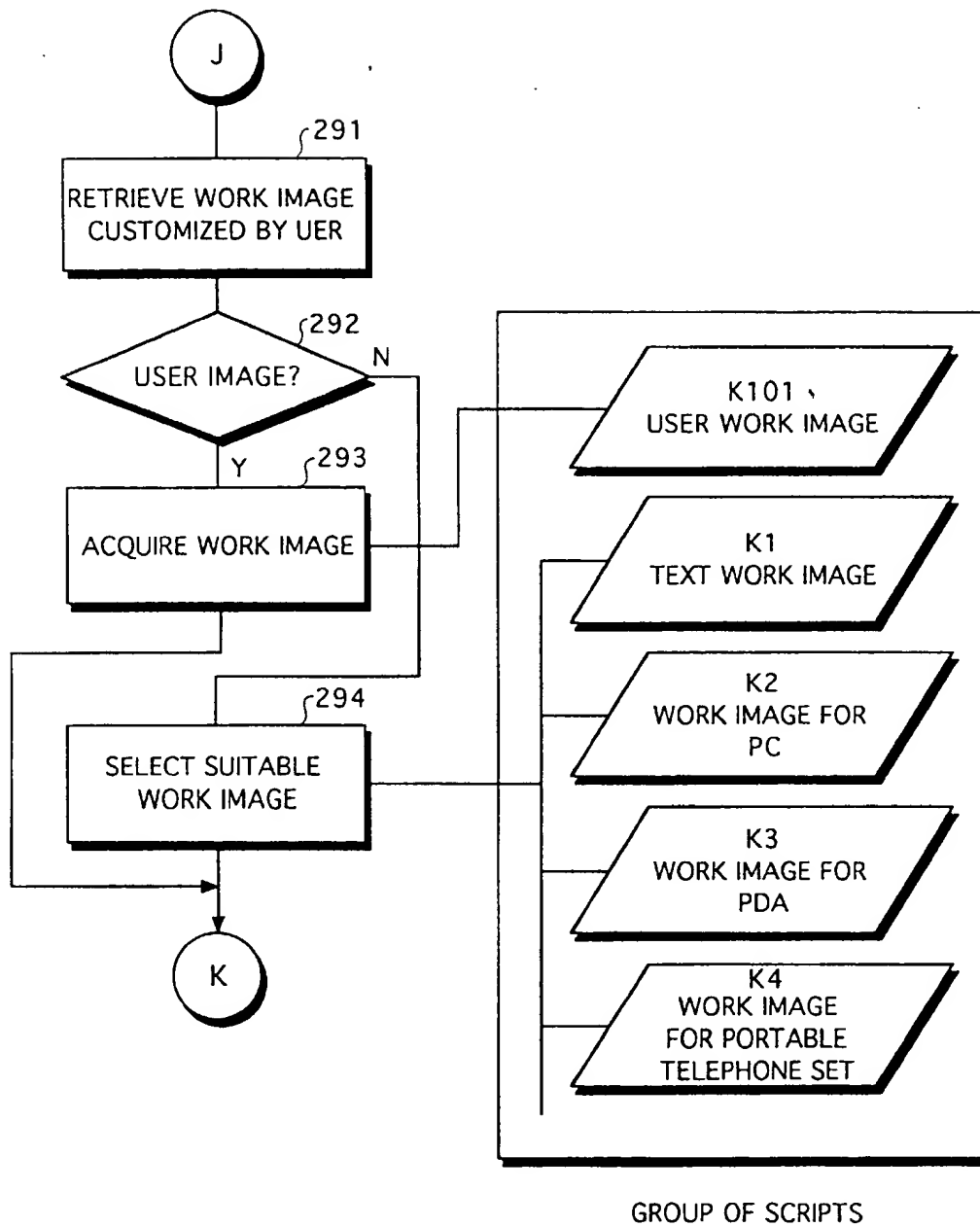


Fig. 31

EDIT IMAGE PRODUCTION PROCESSING

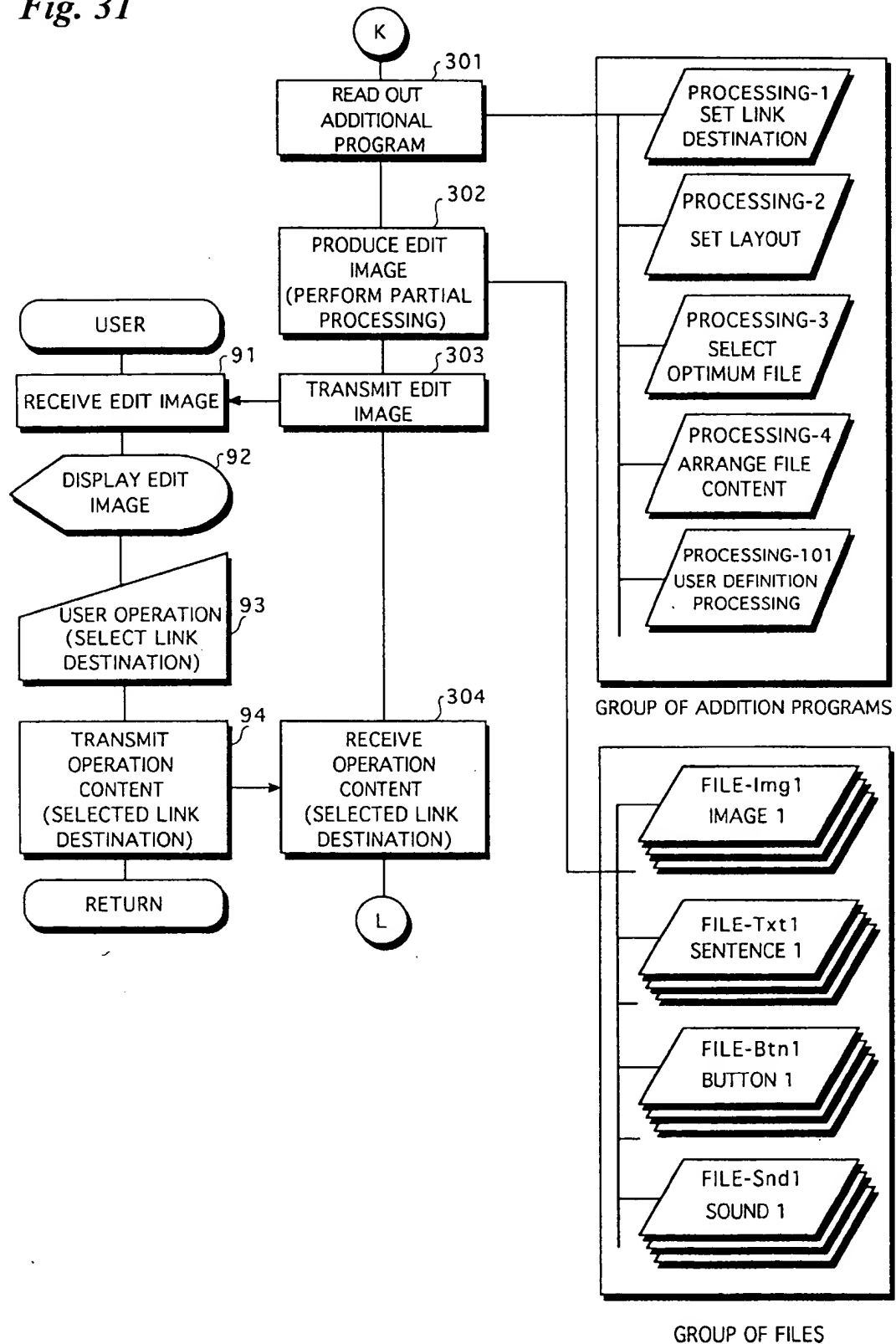


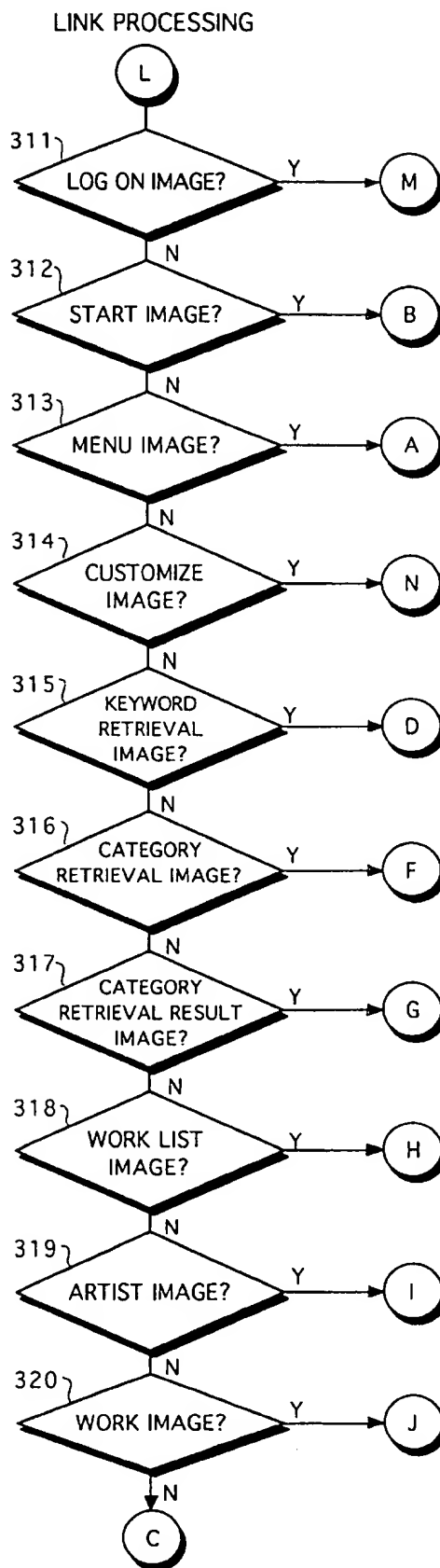
Fig. 32

Fig. 33

CUSTOMIZE MENU PROCESSING

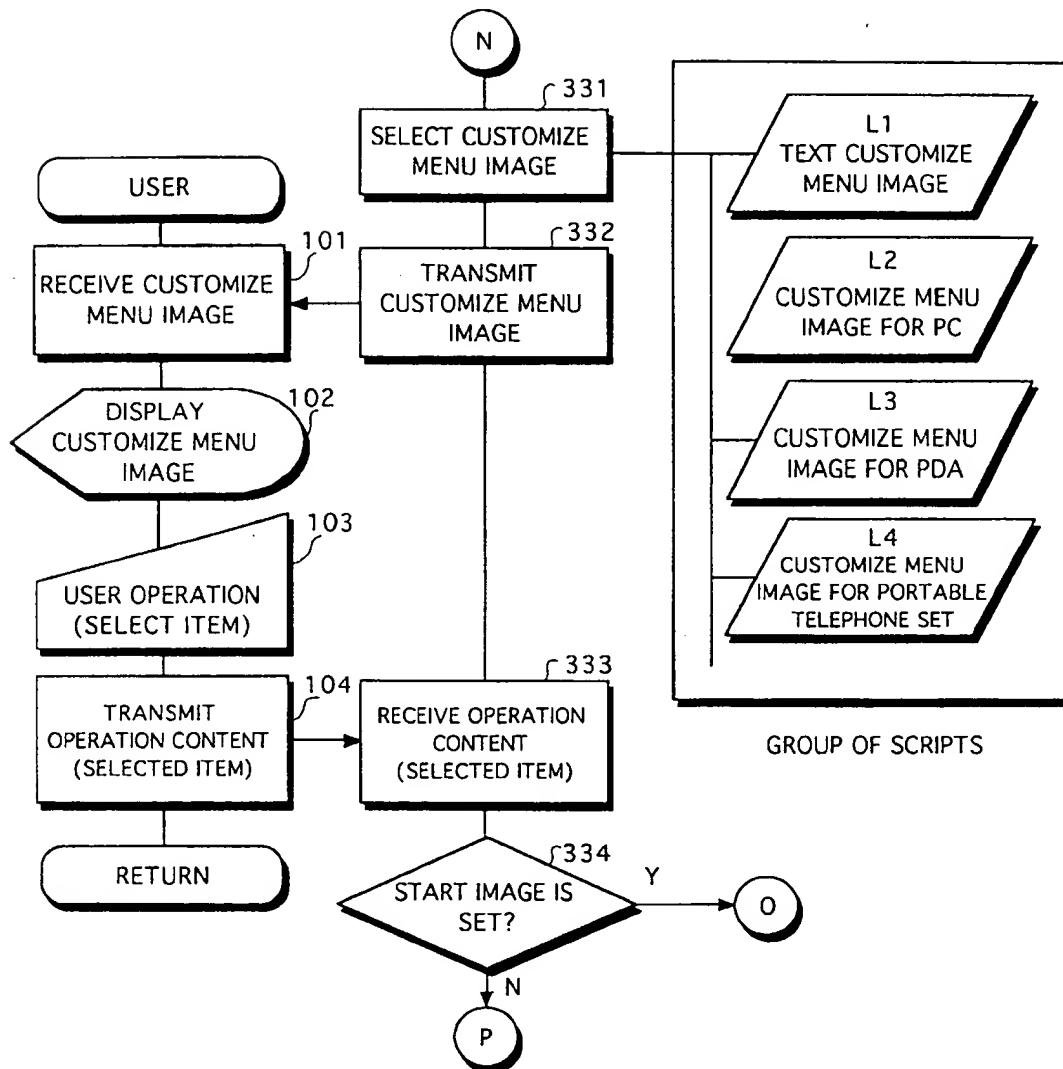


Fig. 34

START IMAGE SETTING PROCESSING

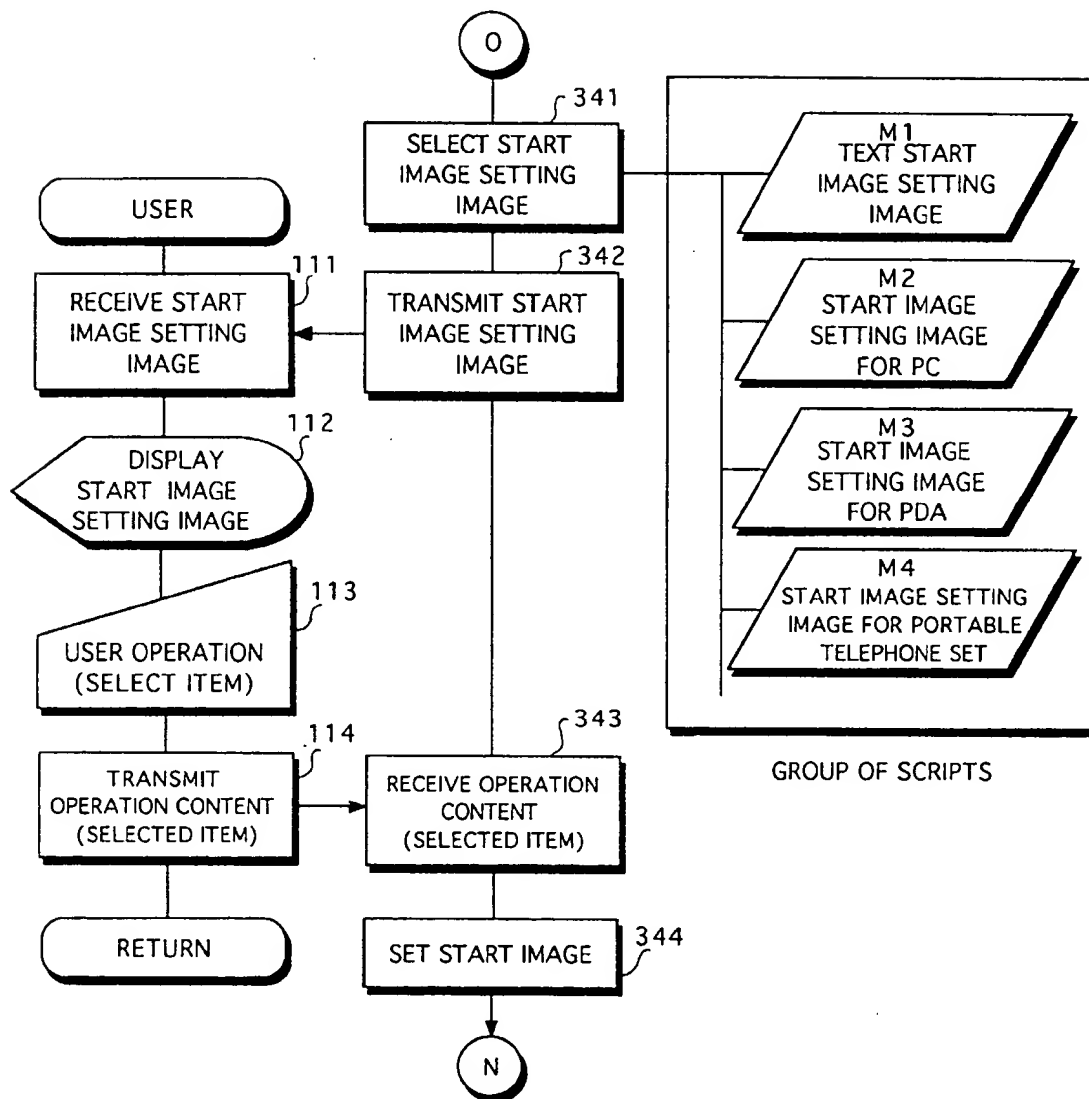


IMAGE EDITING MAIN PROCESSING

Fig. 35

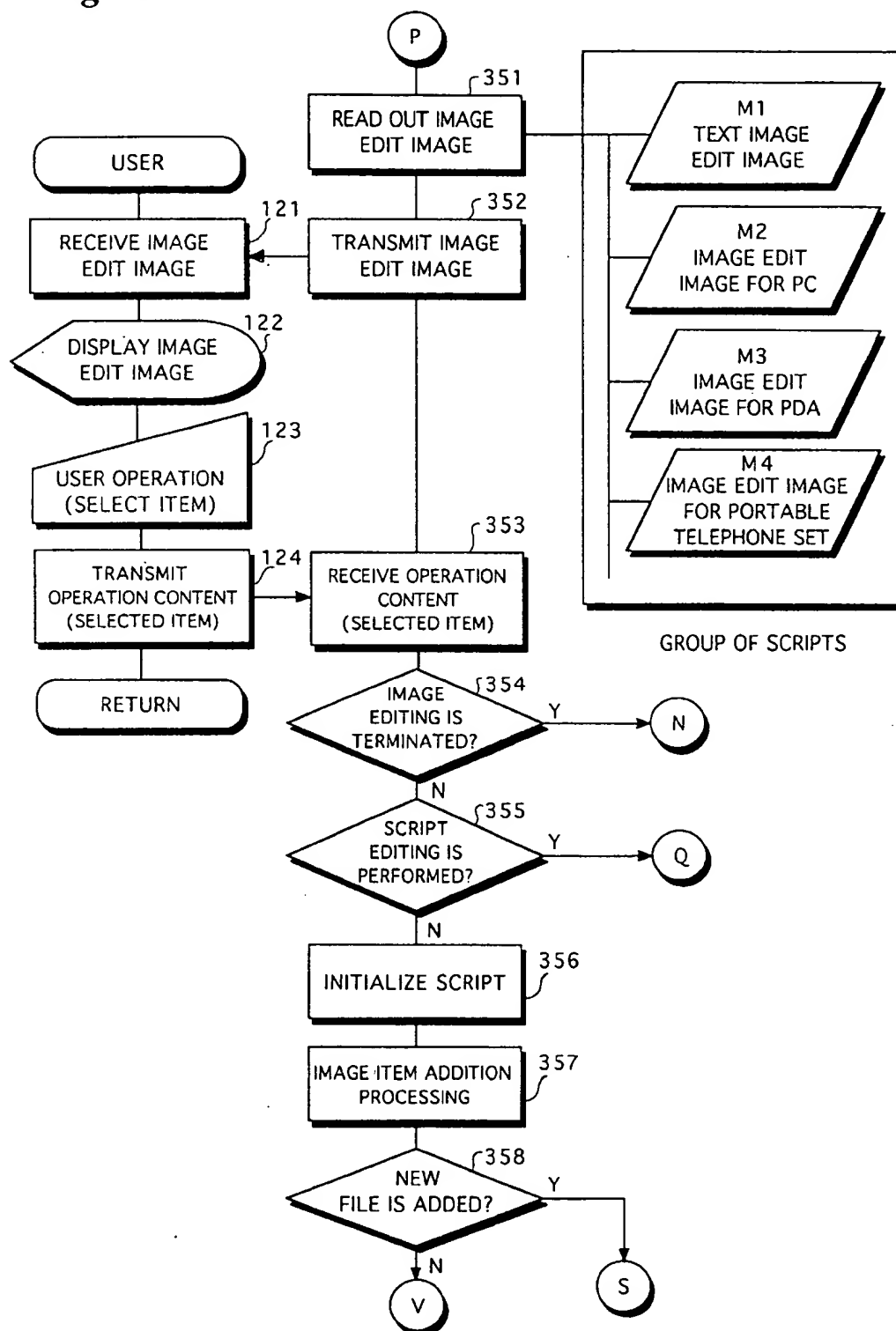


Fig. 36

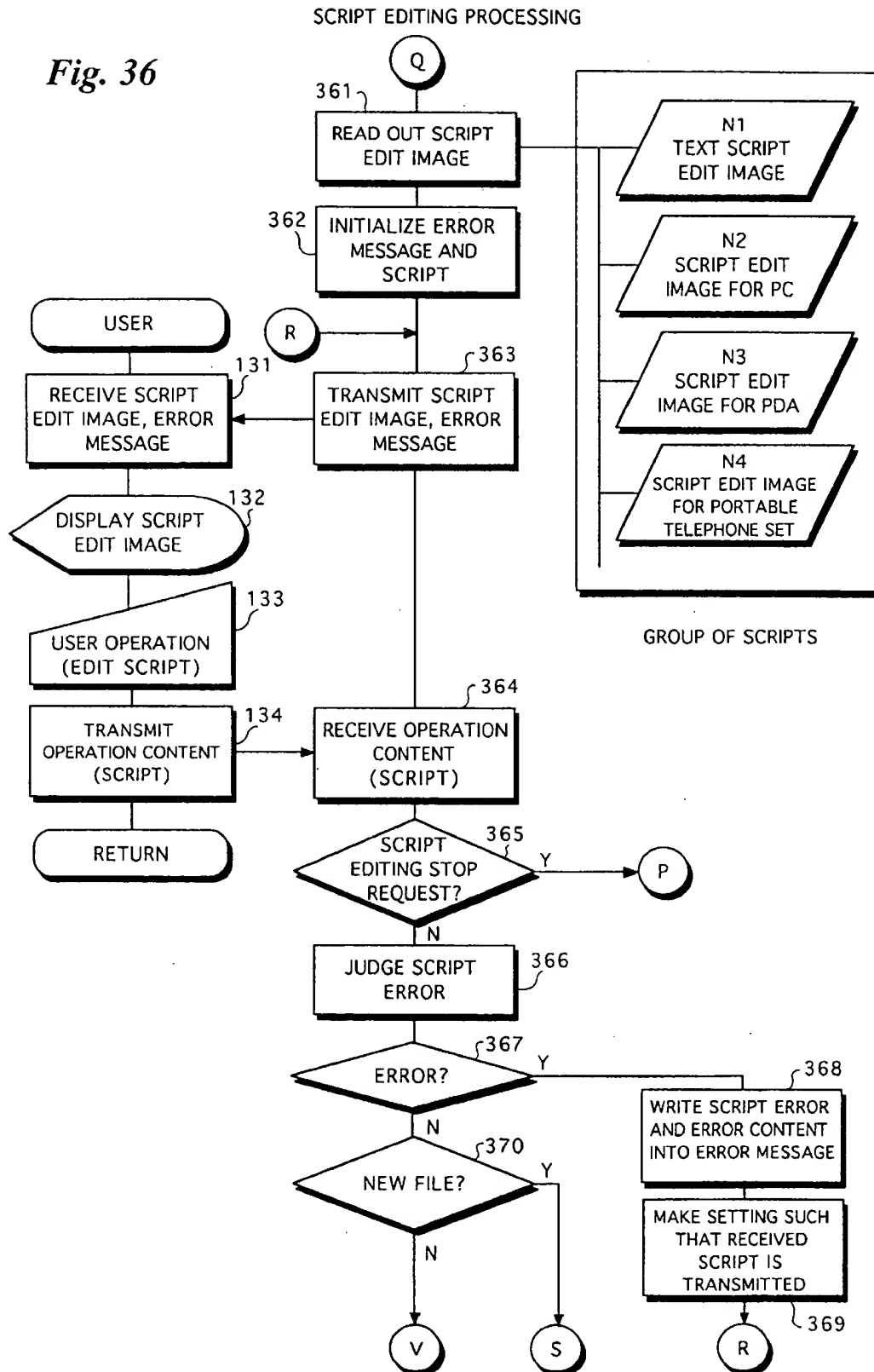


Fig. 37

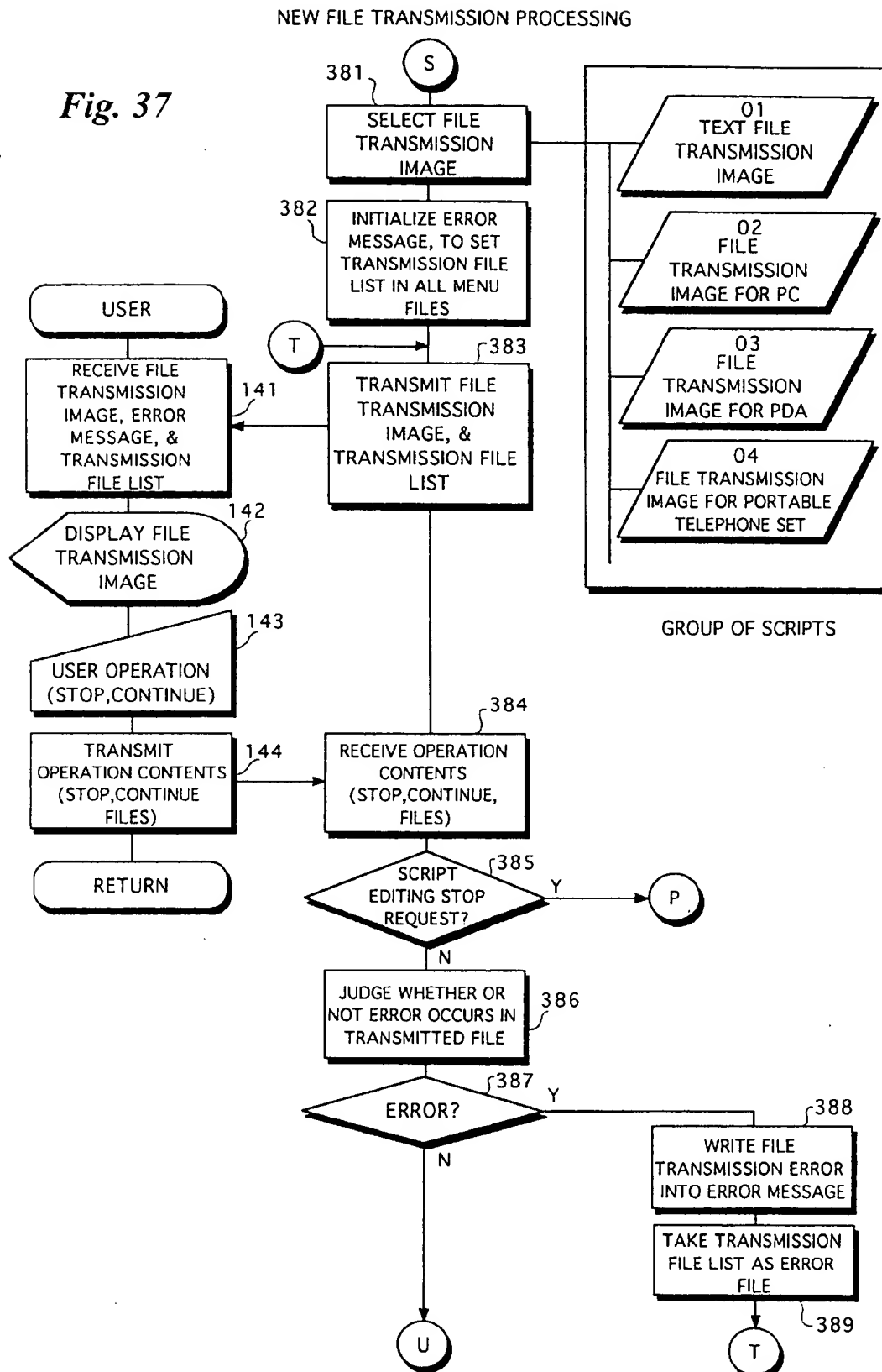


Fig. 38

CUSTOMIZE CONTENT REFLECTION PROCESSING

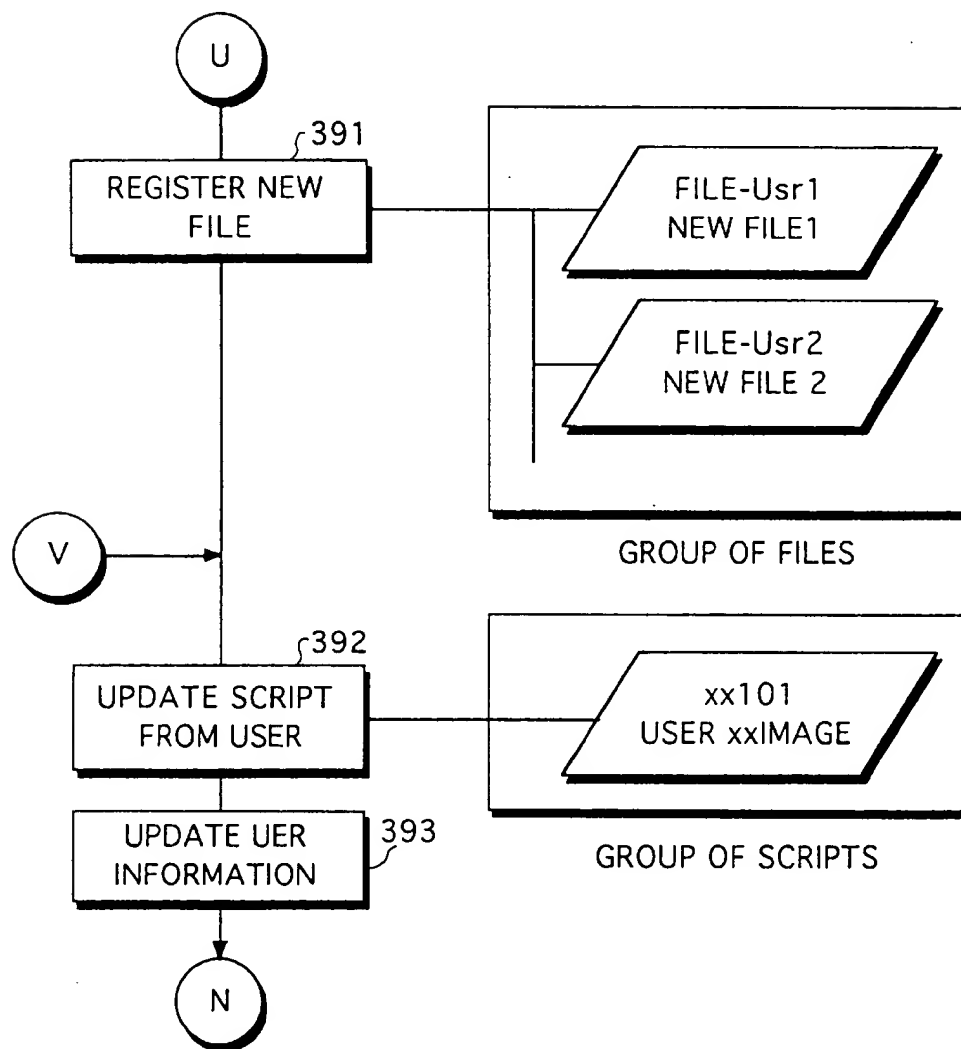


IMAGE ITEM (LAYOUT, FILE) ADDITION PROCESSING

Fig. 39

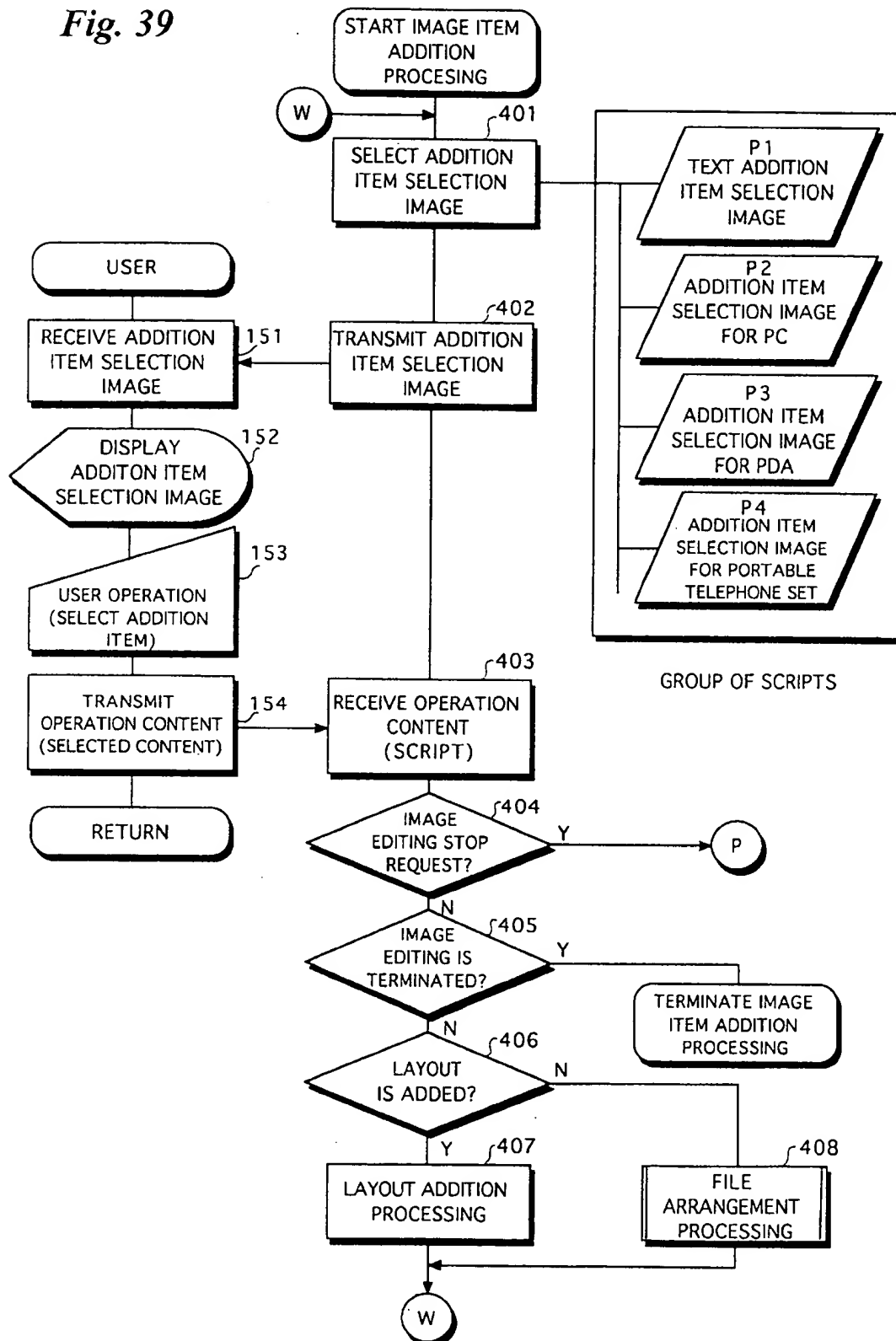


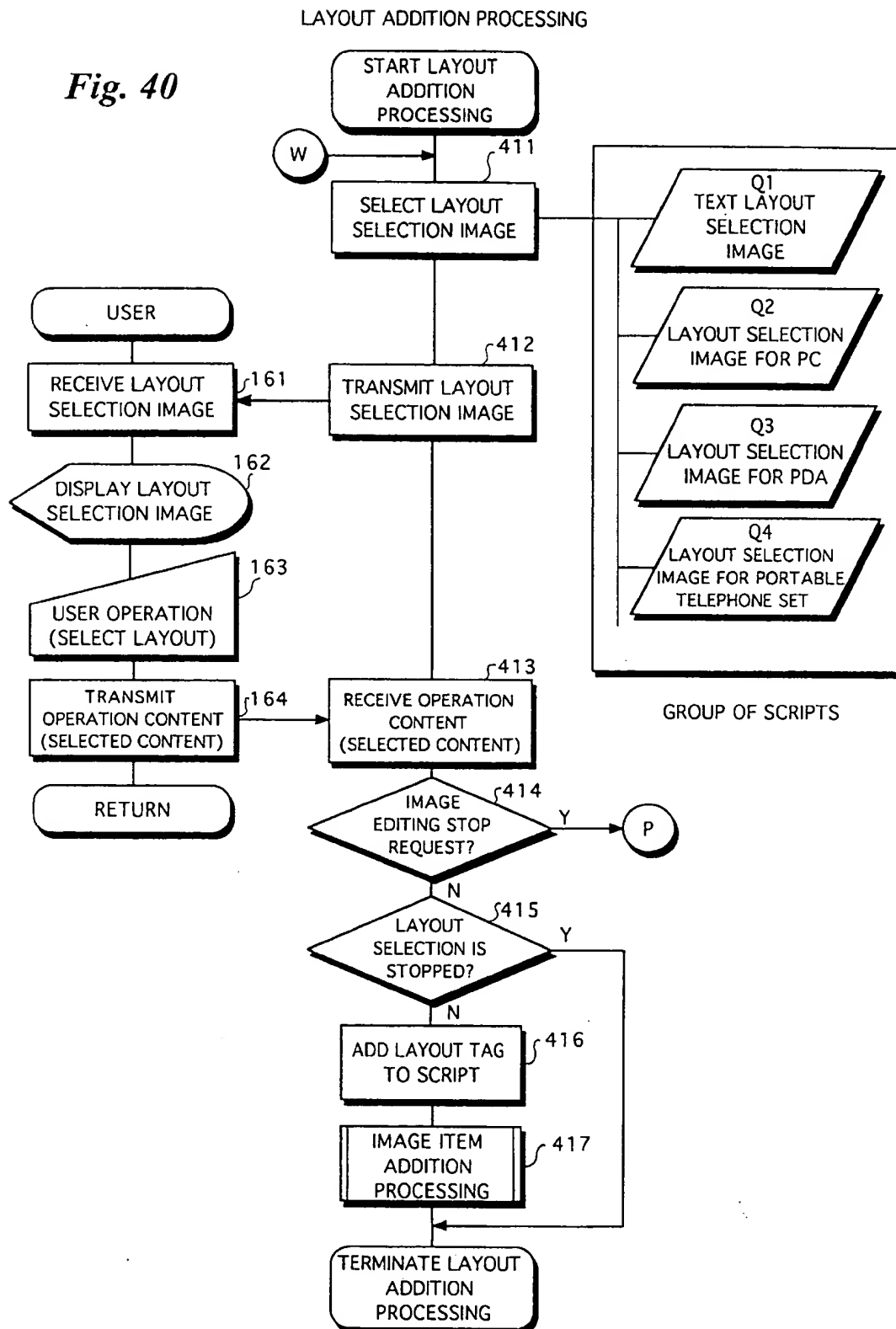
Fig. 40

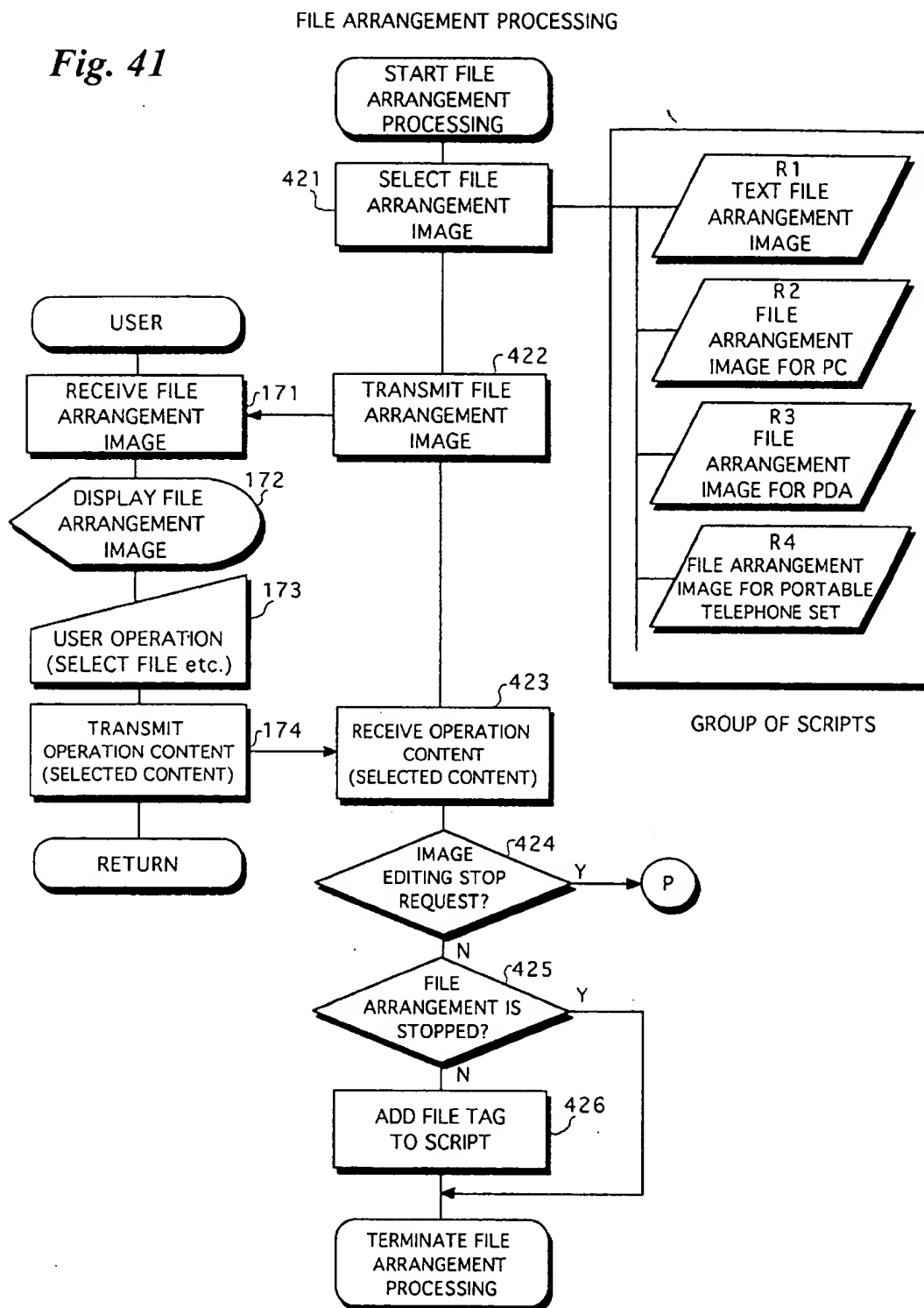
Fig. 41

Fig. 42

LOG ON IMAGE

ENTER USER NAME AND PASSWORD

USER NAME :

PASSWORD :

Fig. 43

MENU IMAGE

SELECT ITEM

1. CUSTOMIZING IMAGE
2. CATEGORY RETRIEVAL IMAGE
3. KEYWORD RETRIEVAL IMAGE

Fig. 44

KEYWORD RETRIEVAL IMAGE

ENTER RETRIEVAL KEYWORD

RETRIEVAL KEYWORD :

Fig. 45

CATEGORY RETRIEVAL IMAGE

SELECT CATEGORY
1. BY ARTIST
2. BY THEME
3. BY YEAR
4. BY COUNTRY
5. BY DATA FORMAT

WORK LIST IMAGE
(PORTABLE TELEPHONE SET)

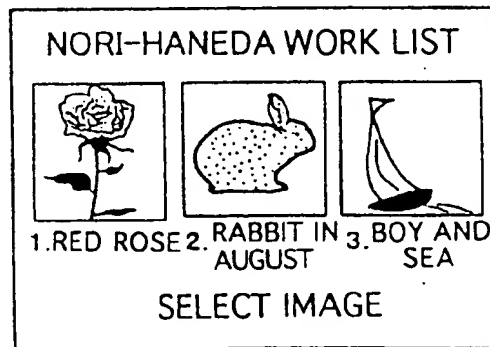
Fig. 46

NORI-HANEDA
WORK LIST
1. RED ROSE
2. RABBIT IN AUGUST
3. BOY AND SEA
SELECT NUMBER

NO IMAGE

WORK LIST IMAGE
PORTABLE INFORMATION TERMINAL (PDA)

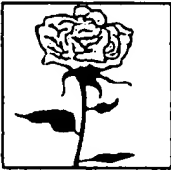
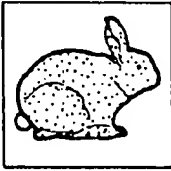
Fig. 47



SMALL IMAGE , BLACK AND WHITE

Fig. 48

WORK LIST IMAGE
PC(SMALL SCREEN)

NORI-HANEDA WORK LIST	
	1. RED ROSE A _____ _____ _____
	RABBIT IN 2. AUGUST B _____ _____ _____
SELECT IMAGE	

MEDIUM-SIZED IMAGE , FULL COLOR
SHORT DESCRIPTIVE TEXT

Fig. 49

WORK LIST IMAGE

LARGE IMAGE, FULL COLOR

LONG DESCRIPTIVE TEXT

TITLE ON THE LEFT, MENU PRESENT

PC (LARGE SCREEN)

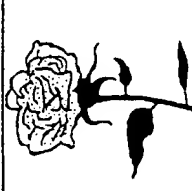
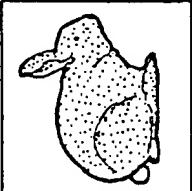
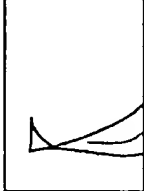
<p>INTERNET IMAGE GALLERY</p>		<p>NORI-HANEDA WORK LIST</p>	
<p>MENU</p>	<p>RETRIEVAL</p>	<p>LIST OF ARTISTS</p>	<p>SELECT IMAGE</p>
<p>1. RED ROSE</p> <p>A</p> 		<p>2. RABBIT IN AUGUST</p> <p>B</p> 	
<p>3. BOY AND SEA</p> <p>C</p> 			

Fig. 50

CUSTOMIZE MENU IMAGE

SELECT IMAGE TO BE CUSTOMIZED

1. START IMAGE
2. MENU IMAGE
3. CUSTOMIZING IMAGE
4. CATEGORY RETRIEVAL IMAGE
5. KEYWORD RETRIEVAL IMAGE
6. WORK LIST IMAGE
7. ARTIST IMAGE

Fig. 51

START IMAGE SETTING IMAGE

SELECT START IMAGE

1. MENU IMAGE
2. CUSTOMIZING IMAGE
3. CATEGORY RETRIEVAL IMAGE
4. KEYWORD RETRIEVAL IMAGE
5. CATEGORY RETRIEVAL RESULT IMAGE
6. WORK LIST IMAGE
7. ARTIST IMAGE

Fig. 52

IMAGE EDIT IMAGE

DO YOU ADD ITEM OR EDIT SCRIPT?

WHEN YOU ADD ITEM, ENTER ITEM TO BE ADDED

Fig. 53

SCRIPT EDIT IMAGE

WE PERFORM BATCH EDITING BY SCRIPT

ENTER SCRIPT

Fig. 54

FILE TRANSMISSION IMAGE

ENTER FILE NAME TO BE TRANSMITTED

Fig. 55

ADDITION ITEM SELECTION IMAGE

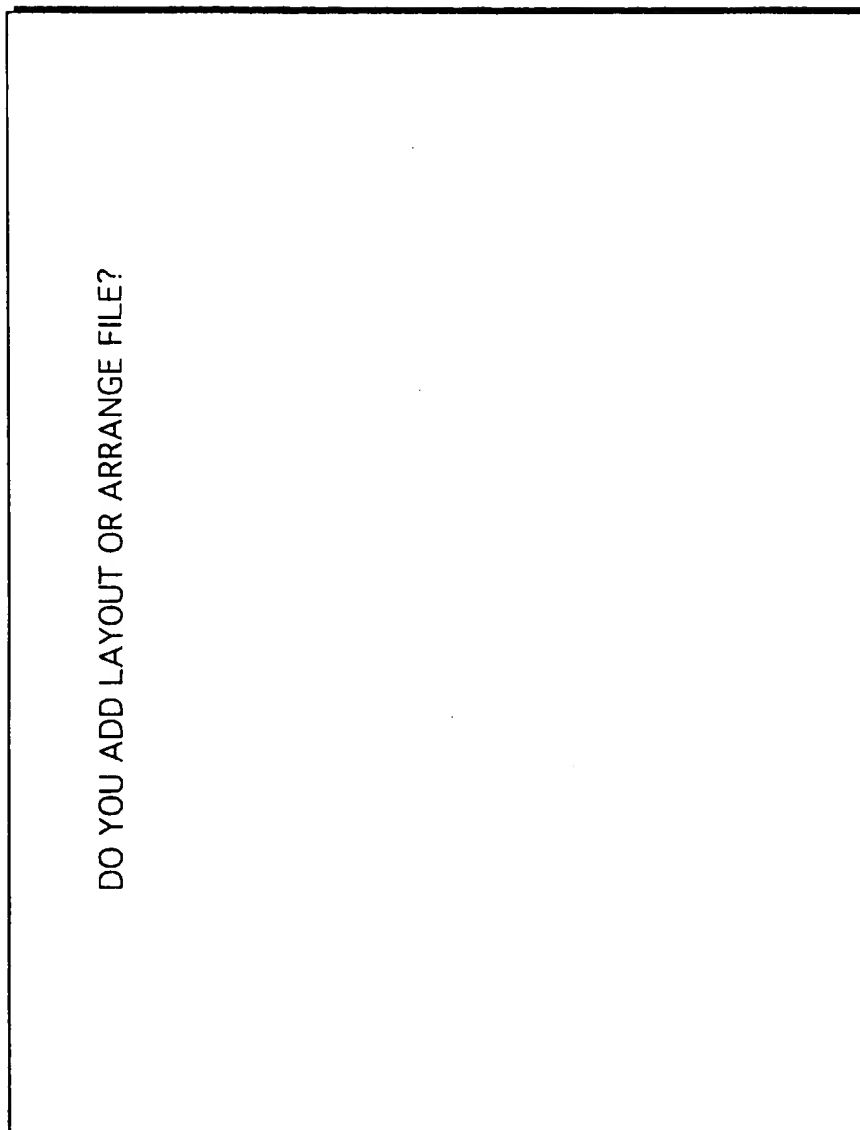


Fig. 56

LAYOUT SELECTION IMAGE

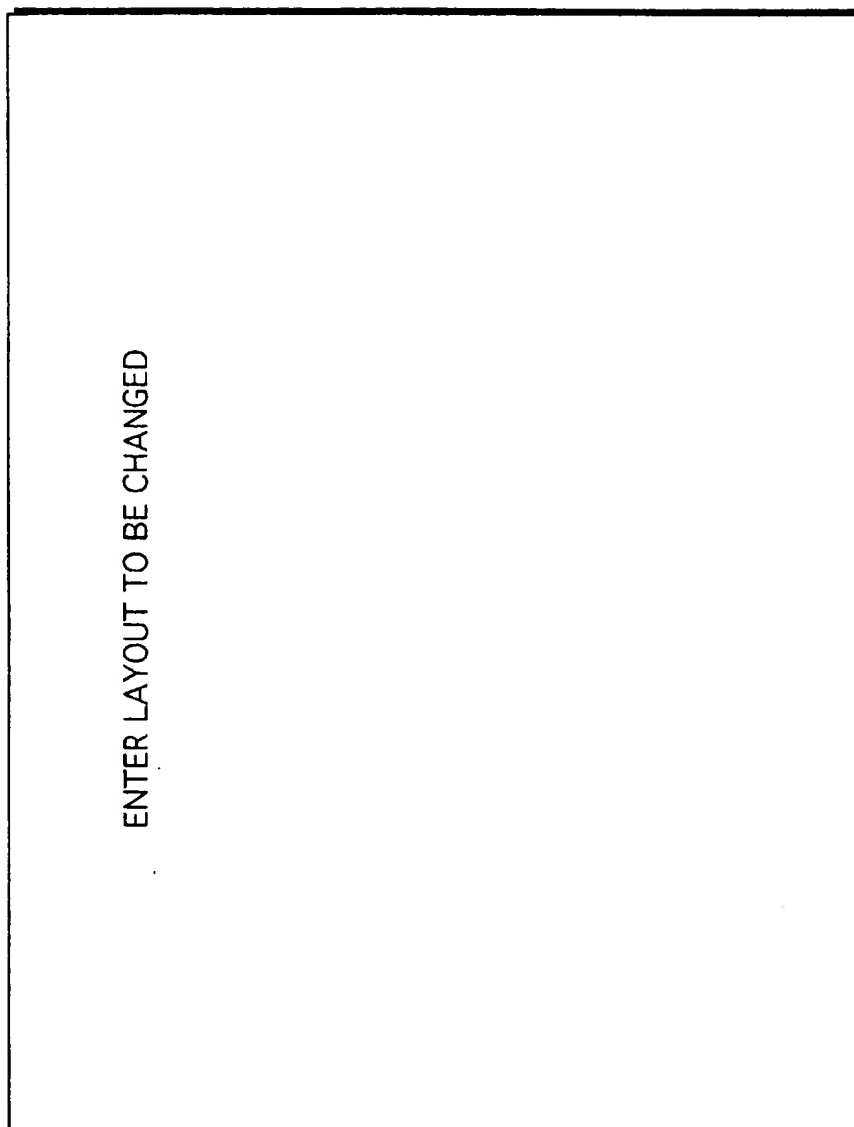


Fig. 57

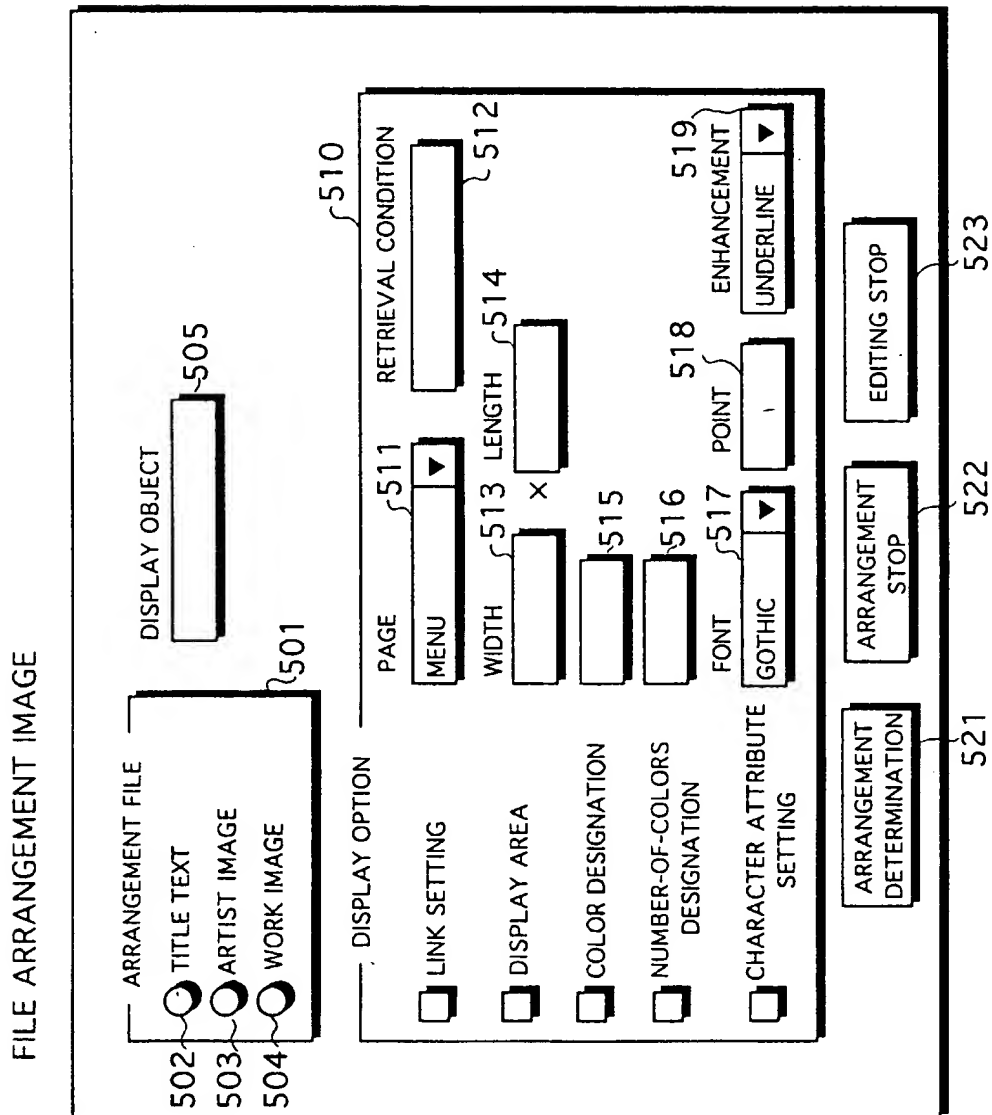


IMAGE COMMUNICATION SYSTEM

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to an image communication system comprising an image transmitting device and an image receiving device which can establish data communication with each other, the image transmitting device constituting the image communication system and an image transmitting method, and a recording medium storing a program for controlling a computer in the image transmitting device.

2. Related Art

With the development of an electronic technique, not only a personal computer but also various devices such as a portable telephone set or a personal digital assistant have allowed data communication. A small-sized display device is also attached to the portable telephone set or the personal digital assistant. It is possible to receive image data transmitted through a network and display an image represented by the received image data on the display device.

However, the display device provided in the portable telephone set, the personal digital assistant, or the like is small. Accordingly, it may, in some cases, be difficult to display the same image as the image displayed on the display device connected to the personal computer.

SUMMARY OF THE INVENTION

An object of the present invention is to make it possible for an image receiving device to receive image data representing an image suitable for an image to be displayed.

The present invention is directed to an image communication system comprising an image transmitting device (apparatus) and an image receiving device (apparatus) which can establish data communication with each other.

The image receiving device (apparatus) comprises device information transmission means (device) for transmitting to the image transmitting device (apparatus) information relating to the image receiving device (apparatus).

The image transmitting device (apparatus) comprises device information receiving means (device) for receiving information relating to the image receiving device (apparatus) which has been transmitted from the device information transmission means (device) in the image receiving device (apparatus), and image data transmission means (device) for transmitting to the image receiving device (apparatus) image data representing an image having a display format suitable for the image receiving device (apparatus) on the basis of the information relating to the image receiving device (apparatus) which has been received by the device information receiving means (device).

The present invention also provides an image transmitting device (apparatus) constituting the image communication system.

The present invention also provides a method suitable for the image transmitting device (apparatus). That is, in an image transmitting device (apparatus) which can establish data communication with an image receiving device (apparatus), the method comprises the steps of receiving information relating to the image receiving device (apparatus) which has been transmitted from the image receiving device (apparatus); and transmitting to the image receiving device (apparatus) image data representing an image having a display format suitable for the image receiving device (apparatus) on the basis of the received information relating to the image receiving device (apparatus).

Furthermore, the present invention also provides a recording medium storing a program for carrying out the above-mentioned method in the image transmitting device (apparatus).

According to the present invention, information relating to the image receiving device (apparatus) (for example, the type of the image receiving device, information relating to display on a display device connected to the image receiving device, i.e., the size of a display screen, the number of colors which can be displayed, software for image display which is installed in the image receiving device, etc.) is transmitted from the image receiving device to the image transmitting device.

In the image transmitting device (apparatus), the information relating to the image receiving device is received. Consequently, image data representing an image having a display format suitable for the image receiving device (an image of a size suitable for the image receiving device, an image in colors whose number is suitable for the image receiving device, the layout of an image in a case where the image is an edit image which is composed of a plurality of images, display items in a case where a plurality of items including an image item are displayed, etc.) is transmitted to the image receiving device on the basis of the received information.

In the image receiving device (apparatus), image data having a format suitable for display is received. The received image data is fed to a display device, so that an image suitable for the display device connected to the image receiving device is displayed.

The image transmitting device (apparatus) may further comprise storage means (device) for storing image data having a plurality of different display formats. In this case, image data representing the image having the display format suitable for the image receiving device out of the storage means will be transmitted to the image receiving device on the basis of the information relating to the image receiving device which has been received by the device information receiving means.

Edit image data representing an edit image which is composed of a plurality of display items such as an image and a sentence (may be data representing the edit image itself, data representing such an image and a sentence that the edit image can be produced in the image receiving device, or data representing a layout for composing the edit image) may be transmitted to the image receiving device.

The image receiving device (apparatus) may further comprise setting means (device) for setting at least one of the layout and the display items of the edit image, and set information transmission means (device) for transmitting to the image transmitting device information set by the setting means.

In this case, the image transmitting device (apparatus) further comprises set information receiving means (device) for receiving the set information which has been transmitted from the set information transmission means in the image receiving device, and determination means (device) for determining at least one of the layout and the display items of the edit image represented by the edit image data on the basis of the set information which has been received by the set information receiving means.

It is possible to display the edit image having a desired layout and display items on the image receiving device.

The setting means in the image receiving device may be at least one of means for performing the setting so as to

3

collectively edit the whole of the edit image and means for performing the setting so as to edit a part of the edit image. Further, the image communication system may further comprise plural image data storage means (devices) storing a plurality of image data representing images composing the edit image in different data amounts (the image data may be thinned to obtain a plurality of image data, or a part of the image may be cut out to obtain a plurality of image data), and production means (device) for producing the edit image using the image represented by any one of the image data stored in the plural image data storage means.

When the image data representing images corresponding to a plurality of frames are transmitted to the image receiving device from the image transmitting device, the image receiving device further comprises edit image designation means for designating the edit image having the layout and the display items at least one of which should be set by the setting means. Further, the setting means sets at least one of the layout and the display item of the edit image designated by the edit image designation means.

It is possible to designate a desired image, and to set at least one of the layout and the display items of the designated image.

The image receiving device (apparatus) can be further provided with selection means (device) for selecting the image first displayed when the image receiving device and the image transmitting device are formally connected to each other, and selected image transmission means (device) for transmitting to the image transmitting device selected image data representing the image selected by the selection means.

The image transmitting device (apparatus) further comprises selected image receiving means (device) for receiving the selected image data which has been transmitted from the selected image transmission means in the image receiving device. In this case, the image data transmission means will transmit image data representing the image represented by the selected image data which has been received by the selected image receiving means to the image receiving device when the image receiving device and the image transmitting device are formally connected to each other.

By selecting a desired image, the selected desired image can be received in the image receiving device when the image transmitting device and the image receiving device are formally connected to each other (connected to each other after it is confirmed that there is an authorization of connection by a password, an ID, or the like).

The image receiving device (apparatus) may further comprise update command transmission means (device) for transmitting an image update command to the image transmitting device.

In this case, the image transmitting device (apparatus) further comprises update command receiving means (device) for receiving the image update command which has been transmitted from the update command transmission means. The image data transmission means will transmit to the image receiving device image data representing the subsequent image which is linked to an image represented by image data which corresponds to the update command received by the update command receiving means and information relating to the image receiving device and has been transmitted to the image receiving means by the image data transmission means.

The image receiving device (apparatus) may further comprise link image setting means (device) for setting the image to be linked, and link information transmission means

4

(device) for transmitting link information set by the link image setting means to the image transmitting device.

At this time, the image transmitting device (apparatus) further comprises link information receiving means (device) for receiving the link information which has been transmitted from the link information transmission means in the image receiving device. Image data representing the subsequent image to be linked is transmitted to the image receiving device on the basis of the link information which has been received by the link information receiving means.

A user can establish a link with a desired image. Images at link destinations differ depending on users.

The foregoing and other objects, features, aspects and advantages of the present invention will become more apparent from the following detailed description of the present invention when taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates the outline of an image communication system;

FIG. 2 is a block diagram showing the electrical configuration of a server;

FIG. 3 illustrates software, etc. stored in the server;

FIG. 4 is a block diagram showing the electrical configuration of a personal computer;

FIG. 5 illustrates software, etc. stored in the personal computer;

FIG. 6 illustrates a transition between images;

FIG. 7 illustrates user information;

FIG. 8A illustrates an artist image management file, and FIGS. 8B to 8D respectively illustrate artist image files;

FIG. 9A illustrates a work image management file, and FIGS. 9B and 9C respectively illustrate work image files;

FIG. 10A illustrates a text management file, and FIG. 10B illustrates a text file;

FIG. 11 illustrates a transition between images;

FIGS. 12 to 19 illustrate examples of a script;

FIG. 20 is a flow chart showing log on processing;

FIG. 21 is a flow chart showing menu image processing;

FIG. 22 is a flow chart showing user start image processing;

FIG. 23 is a flow chart showing error image processing;

FIG. 24 is a flow chart showing keyword retrieval image processing;

FIG. 25 is a flow chart showing retrieval result image processing;

FIG. 26 is a flow chart showing category retrieval image processing;

FIG. 27 is a flow chart showing category retrieval result processing;

FIG. 28 is a flow chart showing work list processing;

FIG. 29 is a flow chart showing artist image processing;

FIG. 30 is a flow chart showing work image processing;

FIG. 31 is a flow chart showing edit image production processing;

FIG. 32 is a flow chart showing link processing;

FIG. 33 is a flow chart showing customize menu processing;

FIG. 34 is a flow chart showing start image setting processing;

5

FIG. 35 is a flow chart showing image editing main processing;

FIG. 36 is a flow chart showing script editing processing;

FIG. 37 is a flow chart showing new file transmission processing;

FIG. 38 is a flow chart showing customize content reflection processing;

FIG. 39 is a flow chart showing image item addition processing;

FIG. 40 is a flow chart showing layout addition processing;

FIG. 41 is a flow chart showing file arrangement processing;

FIG. 42 illustrates a log on image;

FIG. 43 illustrates a menu image;

FIG. 44 illustrates a keyword retrieval image;

FIG. 45 illustrates a category retrieval image;

FIGS. 46 to 49 illustrate work list images;

FIG. 50 illustrates a customize menu image;

FIG. 51 illustrates a start image setting image;

FIG. 52 illustrates an image edit image;

FIG. 53 illustrates a script edit image;

FIG. 54 illustrates a file transmission image;

FIG. 55 illustrates an addition item selection image;

FIG. 56 illustrates a layout selection image; and

FIG. 57 illustrates a file arrangement image.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

(1) Outline of Image Communication System

FIG. 1 illustrates the outline of an image communication system according to the present embodiment.

The image communication system comprises a server 1, a personal computer (PC) 30, a notebook computer 31, a personal digital assistant (PDA) (portable information terminal) 32, and a portable telephone set (portable phone) 33 which can communicate with one another through a network. In the present embodiment, the server 1 is a device on the transmission side, and the personal computer 30, the notebook computer 31, the personal digital assistant 32, and the portable telephone set 33 are user devices on the receiving side. The server 1 which is the device on the transmission side and one of the personal computer 30, the notebook computer 31, the personal digital assistant 32, and the portable telephone set 33 which are the devices on the receiving side may permit data communication with each other.

Data communication is established between the server 1 on the transmission side and the user device on the receiving side and an image is displayed on a display device of the user device on the receiving side. The layout or the like of an image which is displayed on the user device on the receiving side can be changed relatively freely, that is, the image can be customized by a user on the receiving side. The details will be described later.

FIG. 2 is a block diagram showing the electrical configuration of the server 1.

The overall operation of the server 1 is supervised by a CPU 10.

A printer 2 is connected to the server 1. The printer 2 is controlled by a printer control circuit 3.

6

A network card 4 for establishing data communication with the other data communication devices such as the personal computer 30, the notebook computer 31, the personal digital assistant 32 and the portable telephone set 33, an FD drive 5 for recording data on an FD (Floppy Disk) and reading the data recorded on the FD, a CD-ROM drive 6 for reading data recorded on a CD-ROM (Compact Disk-Read Only Memory), a mouse 7, and a keyboard 8 are connected to the server 1. Data obtained from the network card 4 and so forth are accepted in the server 1 through a system I/O controller 9.

A program for executing processing, described later, is stored in the CD-ROM. The program stored in the CD-ROM is read by the CD-ROM drive 6, and is installed in the server 1. It goes without saying that the program may be downloaded from another server or the like.

Furthermore, the server 1 comprises a bus controller 11, a RAM 12 for temporarily storing data, and a ROM 13 storing programs and other necessary data.

Furthermore, there are provided a DVD-RAM drive 15 for recording data on a DVD (Digital Versatile Disk)-RAM (Random Access Memory) or reading data recorded on the DVD-RAM, a tape drive 16 for recording data on a magnetic tape or reading data recorded on the magnetic tape, and a HD (Hard Disk) drive 17 for recording data on a hard disk or reading data recorded on the hard disk. The DVD-RAM drive 15, the tape drive 16, and the HD drive 17 are controlled by an external I/O controller 14.

Furthermore, the server 1 comprises an interrupt controller 18, a timer 19, a memory controller 20, and a VRAM (Video RAM) 21 for temporarily storing image data in order to display an image on the display device 23. The image data stored in the VRAM 21 is fed to a digital analog converter (DAC) 22, so that the image is displayed on a display screen of the display device 23.

FIG. 3 illustrates programs, files, etc. which are stored in a hard disk for the server 1.

The hard disk for the server 1 stores data representing user information, described later, a user information processing program for reading the user information and adding user information, a structured language generation processing program for generating an edit image, a program for transmitting and receiving a file, data representing a script, an addition processing program (although the server 1 has a basic performance for affixing an image in accordance with the script, processing can be further extended (added). A program for the addition (extended) processing is an addition processing program.), and various files such as an image file, a text file and a management file.

FIG. 4 is a block diagram showing the electrical configuration of a personal computer 30. The personal computer 30 has approximately the same configuration as that of the server 1. The same circuits as the circuits constituting the server 1 out of circuits constituting the personal computer 30 are assigned the same reference numerals and hence, the description thereof is not repeated.

The overall operation of the personal computer 30 is supervised by a CPU 40.

A modem 44 is connected to the personal computer 30. The personal computer 30 can be connected to a network by the modem 44.

Furthermore, a flat bed scanner 41, a film scanner 42, and a digital still camera 43 are connected to the personal computer 30. The flat bed scanner 41, the film scanner 42, and the digital still camera 43 are controlled by an external I/O controller 14.

FIG. 5 illustrates programs stored in a hard disk for the personal computer 30.

The programs include a file transmission and receiving program, a structured language display processing program, and a user entry processing program. It goes without saying that other necessary files are stored in the hard disk.

It goes without saying that user devices other than the personal computer 30, that is, the notebook computer 31, the personal digital assistant 32, and the portable telephone set 33 are provided with circuits for data transmission and receiving such that data communication can be established through the network, similarly to the personal computer 30.

FIG. 6 illustrates a transition of images which are displayed on the user device connected to the server 1, that is, a transition of pages in the server 1 which causes the transition of images displayed on the user device.

In the present embodiment, images (pages) are linked to one another and are retrieved, thereby finally displaying a work image of an artist desired by the user on the user display device. FIG. 6 shows that images (pages) which are connected to one another are linked to one another.

In the present embodiment, there are images (pages), described below.

Log on image 50 (see FIG. 42)

It is an image for confirming, when the user accesses the server 1, authorization for the access.

Menu image 51 (see FIG. 43)

It is an image for selecting an image which should be subsequently displayed on the user display device.

Customizing image 52

It is an image for customizing an image by the user.

Category retrieval image 53 (see FIG. 45)

It is an image for performing a category retrieval.

Keyword retrieval image 54 (see FIG. 44)

It is an image for executing a keyword retrieval.

Artist list image 55

It is an image for listing artists whose work images can be seen.

By-theme list image 56

It is an image for listing work images which can be seen by theme (for each theme).

By-year list image 57

It is an image for listing work images which can be seen by year.

By-country list image 58

It is an image for listing work images which can be seen by artist's country.

By-data format list image 59

It is an image for listing work images which can be seen by data format.

Retrieval result list image 60

It is an image for listing results obtained by retrieval.

Work list image 61 (see FIGS. 46 to 49)

It is an image for listing artist's works.

Artist image 62

It is an image for displaying information relating to an artist.

Work image 63

It is an artist's work image which a user desires to obtain.

In the image communication system according to the present embodiment, when the user device accesses the server 1, a log on image is displayed on the user device in order to confirm whether the user has authorization for the access. When it is confirmed that the user has authorization

for the access, the subsequent image (this is referred to as a start image) which is linked to the log on image is displayed on the user display device.

Which of images is taken as the start image is set by the user. If the start image is not set by the user, a menu image is displayed as a predetermined start image on the user display device.

Furthermore, which of images exists at a link destination of the image displayed on the user display device can be also set by the user. For example, when the retrieval result list image 60 is displayed on the user display device, the image at the link destination is the work list image 61 or the artist image 62. However, which of the images exists at a link destination of the retrieval result list image can be determined by the user.

FIG. 7 illustrates an example of user information.

The user information is information relating to the device on the receiving side which is connected to the server 1. The user information is stored in the hard disk for the server 1, as described above. The user information includes data respectively representing a user name, a password, a script for a start image, retrieval conditions, a script for a menu image, a script for a category retrieval image, a script for a work list image, a script for an artist image, and a script for a work image.

The user is previously informed of a user name and a password by mail, for example. A start image or the like is customized by the user. The layout or the like of an image such as a start image is defined by a script.

In the user information shown in FIG. 7, start images by users A to D are set in the category retrieval result image. Therefore, retrieval conditions for obtaining the respective retrieval results are stored in the user information.

FIGS. 8A, 8B, 8C and 8D illustrate a group of artist image files which is stored in the hard disk for the server 1. One ID is assigned to the group of artist image files.

FIG. 8A illustrates a management file of the group of artist image files.

The management file is used when the user accesses the server 1 to retrieve an image which is desired to be read or to be seen (to be reviewed). The management file includes the name of an artist of the image which is desired to be reviewed, the nationality of the artist, the date of birth of the artist, the date of death of the artist, a descriptive text file ID, a file format, the number of image files, and image file names.

FIGS. 8B to 8D respectively illustrate images represented by the image file names stored in the artist image management file shown in FIG. 8A. The image is specified by the image file name stored in the management file.

FIG. 8B illustrates the image having the first image file name in the management file shown in FIG. 8A. The amount of image data representing the image is the largest, and the resolution of the image is high.

FIG. 8C illustrates the image having the second image file name in the management file shown in FIG. 8A. The amount of image data representing the image is medium, and the resolution of the image is medium.

FIG. 8D illustrates the image having the third image file name in the management file shown in FIG. 8A. The amount of image data representing the image is the smallest, and the resolution of the image is low.

The artist image 62 (refer to FIG. 11) can be produced from the artist image management file, the artist image file, and a text file described later.

FIGS. 9A, 9B and 9C illustrate a group of work image files. The group of work image files also has one ID.

FIG. 9A illustrates a work image management file.

The management file is also used when the user accesses the server 1 to retrieve an image which is desired to be reviewed. The work image management file stores the name of an artist, the title of the image, the year of issue of the image, a work category of the image, a descriptive text file ID, a file format, the number of image files, and image file names.

FIGS. 9B and 9C respectively illustrate images represented by the image file names stored in the work image management file shown in FIG. 9A. The image is specified by the image file name stored in the work image management file.

FIG. 9B illustrates the image having the first image file name in the work image management file shown in FIG. 9A. The amount of image data representing the image is large, and the resolution of the image is high.

FIG. 9C illustrates the image having the second image file name in the work image management file shown in FIG. 9A. The amount of image data representing the image is small, and the resolution of the image is low.

The artist image can be produced from the work image management file, the work image file, and a text file described later.

FIGS. 10A and 10B illustrate a group of text files. The group of text files has one ID.

FIG. 10A illustrates a text management file.

The text management file is also used when the user accesses the server 1 to retrieve an image which is desired to be reviewed. The text management file includes a file format, the number of text files, and a text file name.

FIG. 10B is a text represented by the text file having the text file name stored in the text management file shown in FIG. 10A.

When the artist image management file shown in FIG. 8A is found by retrieval, the descriptive text file ID stored in the artist image management file is read. A text file having the read descriptive text file ID is found (the file representing the text shown in FIG. 10B, for example). An artist image shown in FIG. 11, described later, for example, is produced from the found text file and the image management file.

FIG. 11 illustrates a transition of images which are displayed on the user display device.

In the image communication system according to the present embodiment, when the server 1 is first accessed, as described above, the log on image 50 is displayed on the user display device. It is confirmed whether or not the user who accesses the server 1 has authorization by the log on image 50. When it is confirmed that the user has authorization, the start image is displayed on the user display device.

A start image can differ depending on the user, as described above. For example, the start images by the users A and B are the by-year list image 57 (one of images obtained as a result of a category retrieval in accordance with the retrieval conditions in the above-mentioned user information). The start images by the users C and D are the by-theme list image 56 (one of images also obtained as a result of a category retrieval in accordance with the retrieval conditions in the user information).

Furthermore, in the image communication system according to the present embodiment, the link destination of an image can differ depending on the user, as described above. When the user A or B by which the by-year list image 57 is displayed as the start image clicks characters "NORI-HANEDA", for example, the work list image 61 is displayed on the user display device when the user A clicks the characters, while the artist image 62 is displayed on the user display device when the user B clicks the characters.

Similarly, when the characters "NORI-HANEDA" are clicked by the user C or D by which the by-theme list image 56 is displayed as the start image, the work list image 61 is

displayed on the user display device when the user C clicks the characters, while the artist image 62 is displayed on the user display device when the user D clicks the characters.

FIGS. 12 to 19 respectively illustrate examples of scripts for displaying an image layout, a link destination, and so forth.

FIGS. 12 and 13 respectively illustrate scripts for displaying the by-year list image 57 shown in FIG. 11. FIG. 12 illustrates a script about the user A, and FIG. 13 illustrates a script about the user B. In the scripts, "function1" represents a link destination. In FIG. 12, function1 (a work list . . .) is shown. Accordingly, a link from the by-year list image 57 to the work list image 61 is established. In FIG. 13, function1 (artist . . .) is shown. Accordingly, a link from the by-year list image 57 to the artist image 62 is established.

FIGS. 14 and 15 respectively illustrate scripts for displaying the by-theme list image 56 shown in FIG. 11. FIG. 14 illustrates a script about the user C, and FIG. 15 illustrates a script about the user D. With respect to the user C, a link from the by-theme list image 56 to the work list image 61 is established. With respect to the user D, a link from the by-theme list image 56 to the artist image 62 is established.

FIG. 16 illustrates scripts for displaying the work list image on a display device in the portable telephone set 33, FIG. 17 illustrates a script for displaying the work list image on a display device in the personal digital assistant (portable information terminal) 32, FIG. 18 illustrates a script for displaying the work list image (the work list image shown in FIG. 11) on a display device in the notebook computer 31, and FIG. 19 illustrates a script for displaying the work list image on a display device in the personal computer 30.

In the scripts respectively illustrated in FIG. 12 to FIG. 19, mod indicates an operator for finding a remainder, Txt15 indicates a title text file ID "a list of contemporary artists", Txt16 indicates a title text file ID "a list of animal images", function1 indicates a link with a page of an argument 1, an argument 2 indicates retrieval conditions (a character string marked off by a comma in small parentheses of function is an argument. When there are a plurality of arguments, the first argument is an argument 1, and the subsequent argument is an argument 2), function3 indicates display using a font for the argument 1, function4 indicates display of a text using a point of an argument, function5 indicates display of characters, whose number corresponds to the number of characters composing the argument 1, from the head in a size two times the original size, function6 indicates display using characters whose number is within the number of characters composing the argument 1, function10 indicates display in colors whose number corresponds to the number of colors of the argument 1, function11 indicates display in a size corresponding to the size of the argument 1 by the size of the argument 2, function21 indicates that the color of the argument 1 is a foreground color and the color of the argument 2 is a background color, function31 indicates framing with a line width of an argument, and function32 indicates underlining with a line width of the argument 1.

Furthermore, a portion enclosed by < > is interpreted as a tag. A layout and a file arrangement are realized by the tag.

The layout includes multiple column and alignment.

The multiple column is realized by the following formats:

```
<begin column 1>
(content in the first column)
<end column 1>
<begin column 2>
(content in the second column)
<end column 2>
<begin column N>
(content in the N-th column)
<end column N>
```

When the column 1 is produced in the course of the multiple column, the multiple column is nested (has a hierarchical structure).

```
<begin column 1>
<begin column 1>
:
<end column 1>
<begin column 2>
:
<end column 2>
<end column 1>
```

The alignment includes left justification, centering, and right justification, and is realized by the following formats:

```
<begin left>
:
(left justification)
:
<end left>
<begin center>
:
(centering)
:
<end center>
<begin right>
:
(right justification)
:
<end right>
```

Furthermore, a display area is produced over a plurality of columns in the following manner.

```
<begin column 1>
:
<region #1,120,80>
:
<end column 1>
<begin column 2>
<... using region #1>
:
<end column 2>
```

In the course of the column 1, a display area having region ID=#1 is ensured in a size of 120×80, and a display area which is coupled to the region is used at the beginning of the column 2.

In order to acquire the content of the management file in the script, a value shall be accepted in a variable in the following manner using a manage tag.

```
#A= <manage (Img1, image title)>
```

An image title having file ID=Img1 is substituted in the variable #A.

Furthermore, when an attempt to arrange the management file in the script is made, the following shall be describable.

```
<file manage (Img1, image title)>
```

Not an image having file ID=Img1 but the title of the image (a text) is arranged in the script.

A character string, starting with "#", shall be a variable herein. Further, in order to reflect the result of retrieval on the script, the following retrieval variables are defined:

```
##Count
the number of cases which have hit retrieval
##FileID( )
the file ID arrangement of a file which has hit retrieval
##SearchCondition
retrieval conditions to be handed to a page at a link
destination
```

For example,

```
##SearchCondition="File format=artist and date of
birth≥1950/1/1"
```

In order to perform repetitive processing or the like, a control tag is defined.

```
If ... then ... else ... endif tag
```

This is an interpretive tag with conditions. It is for interrupting a script in a predetermined range only when it coincides with the conditions.

```
for ... to ... next tag
```

This is a repeated interpretive tag.

The arrangement of a file is basically realized by designating a file tag and a file ID. Processing can be added, as required.

Processing is added by adding a function parameter to the file tag. An argument can be described in the function parameter. When a plurality of functions, they are interpreted rightward from the left.

For example,

```
<file Img1>
```

It means the arrangement of an (image) file having file ID=Img1.

```
<file Txt32 function 3(2)>
```

It means the arrangement of a (text) file having file ID=Txt32. In the case, processing designated by function 3 is performed. For example, a frame having a thickness 2 is added. An example is display in a color assigned a color number 2.

```
<file Btn12 function1 (I101)>
```

It means the arrangement of a (button) file having file ID=Btn12. When function 1 shall be setting of a link, a link with I101 ("work list image" defined by the user) is executed.

```
<file Img2 function4 (400,300) function7 (256)>
```

In the case of such processing that function 4(x,y) means a size of x x y and function 7(i) means i colors, an image closest to 400×300 is selected as an image file Img2. If there is no image in colors whose number is not more than 256, the image is transmitted to the display side after it is subjected to color reduction processing.

(2) Communication Processing

FIGS. 20 to 41 are flow charts showing the procedure for communication processing (retrieval of a desired work image) between the server 1 and the user device. FIGS. 42 to 57 respectively illustrate examples of images displayed on the display device connected to the user device. In FIGS. 42 to 57, an image represented by text data is illustrated, except in FIGS. 47 to 49 and 57. When the user device can display an image other than the text (an image representing a picture), however, an image including the picture or the like is displayed.

I. Log on Processing

Log on processing is first performed.

A request to transmit a log on image is entered by the user device (step 51). Consequently, the log on image request, browser information (which browser is installed, or whether or not no browser is installed) in the user device, and information relating to the user device (for example, the number of colors which can be displayed on the display device connected to the user device, the size of an image which can be displayed, etc.) are transmitted to the server 1 from the user device (step 52).

13

The log on image request, the browser information, the device information which have been transmitted from the user device are received by the user 1 (step 201). Image data representing a suitable log on image which can be displayed on the display device connected to the user device is selected from the hard disk for the server 1 on the basis of the browser information and the device information which have been transmitted from the user device (step 202). The hard disk for the server stores image data respectively representing a text log on image, a log on image for a PC (Personal Computer) (a log on image for a personal computer and a log on image for a notebook computer are stored, as required), a log on image for a PDA (Personal Digital Assistant), and a log on image for a portable telephone set. Suitable image data which conforms to the user device is selected in the server 1. Image data representing the selected log on image is transmitted to the user device from the server 1 (step 203).

The image data representing the log on image which has been transmitted from the server 1 is received in the user device (step 53). The received image data is fed to the display device, where the log on image is displayed (step 54, see FIG. 42).

FIG. 42 illustrates an example of a text log on image. When no browser is installed in the user device, the text log on image is displayed. In the text log on image, an instruction to enter a user name and a password is displayed. The user enters his or her own name and password in accordance with the instruction (step 55). The user name and the password which have been entered are transmitted to the server 1 from the user device (step 56).

If the log on image is for a portable telephone set, a slight ornament such as clip art is attached to the log on image. If the log on image is for a PDA, an ornament is further attached to the log on image for a portable telephone set. If the log on image is for a PC, an ornament is further attached to the log on image for a PDA. The log on image suitable for the device which has accessed the server (suitable for the browser) is displayed on the display device connected to the user device.

In the server 1, the user name and the password which have been transmitted from the user device are received (step 204). User information stored in the hard disk for the server 1 is referred to. It is checked whether or not the user name and the password which have been transmitted from the user device are stored in the user information, that is, it is checked whether or not there are a user name and a password in the user information which coincide with the user name and the password which have been transmitted or, it is checked whether or not the transmitted user name and the transmitted password have a predetermined relation (step 205).

If there are no user name and no password in the user information which coincide with the transmitted user name and the password (NO at step 206), image data representing a text log on failure image stored in the hard disk for the server 1 is read out. The image data read out is transmitted to the user device from the server 1 (step 207).

In the user device, image data representing the log on failure image is received (step 57). The image data is fed to the display device connected to the user device. Consequently, the log on failure image is displayed on the display device connected to the user device (step 58).

In the image communication system according to the present embodiment, the user can set a start image displayed subsequently to the log on image on the display device connected to the user device. When there are a user name

14

and a password in the user information which coincide with the transmitted user name and the password (YES at step 206), therefore, user information (see FIG. 7) is referred to, and it is checked whether or not the start image of the user which has been logged on has been set by the user (step 209).

When the start image has been set (YES at step 209), the program proceeds to user start image processing shown in FIG. 22.

II. Start Image Processing

In the start image processing, the user information is referred to, so that start image information representing a start image of a user who accesses the server 1 (information stored in a start image item in the user information shown in FIG. 7) is read out (step 219). When the start image information representing the start image is read out, the program proceeds to link processing shown in FIG. 32. It is checked which of images is the start image on the basis of the start image information read out of the user information (steps 311 to 320). The set image is displayed subsequently to the log on image as the start image on the display device connected to the user device.

If the start image coincides with none of the images, the program proceeds to error image processing shown in FIG. 23, assuming that an error occurred.

III. Error Image Processing

In the error image processing, an error image which is suitable for the user device and the browser is selected out of image data representing error images which are recorded on the hard disk for the server 1 (step 221). The error image is produced in the server 1, and data representing the error image is transmitted to the user device from the server 1 (step 222).

When the data representing the error image which has been transmitted from the server 1 is received in the user device (step 71), data representing the error image is fed to the display device connected to the user device. The error image is displayed on the user display device (step 72).

The error image includes a menu button. When the menu button is selected by the user (step 73), data indicating that the menu button has been selected is transmitted to the server 1 from the user device (step 74).

When the data indicating that the menu button has been pressed is received in the server 1 (step 223), the program proceeds to menu image processing shown in FIG. 21.

Unless the start image has been set by the user (NO at step 209 in FIG. 20), the program proceeds to the menu image processing shown in FIG. 21.

IV. Menu Image Processing

In the image communication system according to the present embodiment, an image to be displayed on the user display device can be customized by the user himself or herself, as described above. A menu image is retrieved (step 211), so that it is checked whether or not the menu image is customized by the user. When "No" is described on the user information shown in FIG. 7, it is indicated that the menu image is not customized by the user. When the menu image is not customized, a predetermined image is displayed.

When the menu image is not customized by the user (No at step 212), data suitable for the user device which accesses the server 1 and the browser out of data representing a predetermined menu image is read out of the hard disk for

15

the server 1 (step 214). The data representing the menu image read out is transmitted to the user device from the server 1 (step 215).

When the menu image is customized by the user (YES at step 212), data representing the customized menu image (a user image) is read out of the hard disk for the server 1. Data representing the menu image read out is transmitted to the user device from the server 1 (step 215).

Even when the menu image is customized by the user, the server 1 may, in some cases, be accessed by a device which cannot display the customized menu image. In the case, data which can be displayed by the user device (for example, text data) out of the data representing the customized menu image is transmitted to the user device from the server 1.

When the data representing the menu image which has been transmitted from the server 1 is received in the user device (step 61), the data representing the menu image is fed to the user display device. On the user display device, the menu image is displayed, as shown in FIG. 43 (step 62).

In the menu image, items "1. Customizing Image", "2. Category Retrieval Image", "3. Keyword Retrieval Image" are displayed. The user selects the image (item) to be displayed on the display device out of the items (step 63). Data representing the selected item is transmitted to the server 1 from the user device (step 64).

When the data representing the item which has been transmitted from the user device is received in the server 1 (step 216), it is checked which of the images "1. Customizing image", "2. Category Retrieval Image", and "3. Keyword Retrieval Image" needs to be displayed (steps 217 and 218).

If the user selects "1. Customizing image" (YES at step 217), customize menu processing shown in FIG. 33 is performed.

V. Customize Menu Processing

Referring to FIG. 33, data representing a customize menu image which is suitable for the user device and the browser is read out of the server 1 (step 331). The data read out is transmitted to the user device from the server 1 (step 332).

When data representing the customize menu image which has been transmitted from the server 1 is received in the user device (step 101), the customize menu image is displayed, as shown in FIG. 50, on the user display device (step 102). An instruction to select an image to be customized by the user is displayed on the customize menu image. Examples of the image which can be customized include "1. Start Image (which of images is taken as a start image, that is, an image displayed subsequently to the log on image is set)", "2. Menu Image", "3. Customizing Image", "4. Category Retrieval Image", "5. Keyword Retrieval Image", "6. Work List Image", and "7. Artist Image". The user selects the image to be customized out of the images (items) (step 103). Data representing the selected image (item) is transmitted to the server 1 from the user device (step 104).

When data representing the item selected by the user is received in the server 1 (step 333), it is checked whether or not the item indicates the setting of the start image (step 334).

If the item selected by the user indicates the setting of the start image (YES at step 334), the program proceeds to start image setting processing shown in FIG. 34.

VI. Start Image Setting Processing

Referring to FIG. 34, a start image setting image which is suitable for the user device and the browser is read out of the

16

hard disk for the server 1 (step 341). Data representing the start image setting image read out is transmitted to the user device from the server 1 (step 342).

When the data representing the start image setting image which has been transmitted from the server 1 is received in the user device (step 111), the data representing the start image setting image is fed to the user display device. The start image setting image is displayed on the user display device (step 112).

FIG. 51 illustrates an example of the start image setting image displayed on the user display device.

An instruction to select a start image and items of selectable start images are displayed on the start image setting image. In the present embodiment, "1. Menu Image", "2. Customizing image", "3. Category Retrieval Image", "4. Keyword Retrieval Image", "5. Category Retrieval Result Image", "6. Work List Image", and "7. Artist Image" can be selected as the start image. The items are displayed on the start image setting image. Any one of the items is selected by the user (step 113). Data representing the selected item is transmitted to the server 1 from the user device (step 114).

In the server 1, data representing the selected item which has been transmitted from the user device is received (step 343). The start image is written into the corresponding start image item in the user information (step 344). When the user A selects "5. Category Retrieval Result Image", for example, as the start image, "Category Retrieval Result Image" is written into the start image item in the user information shown in FIG. 7. Thereafter, the program is returned to the customize menu image processing shown in FIG. 33.

In a case where a customize menu image shown in FIG. 50 is displayed on the user display device, when the item other than the customizing of the start image is selected (NO at step 334), the program proceeds to image editing main processing shown in FIG. 35. The image editing main processing is for judging whether image editing is performed by adding an item or by batch script editing.

VII. Image Editing Main Processing

In the image editing main processing, data representing an image edit image which is suitable for the user device and the browser is read out in the server 1 (step 351). Data representing the image edit image read out is transmitted to the user device from the server 1 (step 352).

When the data representing the image edit image which has been transmitted from the server 1 is received in the user device (step 121), the data representing the image edit image is fed to the user display device (step 122). The image edit image as shown in FIG. 52 is displayed on the user display device (step 123). "Do you add item or do you edit script? When you add item, enter item to be added" is displayed on the image edit image.

The user enters an instruction to perform either item addition or script editing into the user device in accordance with the display on the image edit image. If an item is to be added, the item to be added is entered (step 124). Selection data representing the addition of the item selected by the user (and the name of the item to be added) or the script editing is transmitted to the server 1 from the user device (step 124).

When the selection data which has been transmitted from the user device is received in the server 1 (step 353), it is judged whether image editing is terminated (step 354), script editing is performed (step 355), and item addition is performed.

17

If the image editing is terminated (the termination of the image editing can be designated by the user, although it is omitted in FIG. 52) (YES at step 354), the program proceeds to the customize menu image processing shown in FIG. 33.

If the script editing is performed (YES at step 355), the program proceeds to script editing processing shown in FIG. 36.

VIII. Script Editing Processing

Referring to FIG. 36, if the script editing processing is performed, data representing a script edit image which is suitable for the user device and the browser is read out of the hard disk for the server (step 361). Further, a script for an edit image which the user attempts to customize is initialized (step 362). The data read out is transmitted to the user device from the server 1 (step 363).

When the data representing the script edit image which has been transmitted from the server is received in the user device (step 131), the data representing the script edit image is fed to the user display device. The script edit image as shown in FIG. 53 is displayed on the user device (step 132).

Characters "We will perform batch editing by script. Enter script." are displayed on the script edit image. When a script is entered by the user (step 133), the entered script is displayed on the script edit image. Data representing the script entered by the user is transmitted to the server 1 from the user device (step 134).

When the data representing the script which has been transmitted from the user device is received in the server 1 (step 364), it is judged whether or not a request to stop script editing is issued (step 365) (the stop of script editing can be also entered by the user, although it is omitted in FIG. 53).

If the script editing stop request is issued (YES at step 365), the program is returned to the image editing main processing shown in FIG. 35.

If the script editing stop request is not issued (NO at step 365), it is checked whether or not an error occurs in the transmitted data representing the script (steps 366 and 367).

If an error occurs (YES at step 367), an error message is taken as a script error, and the content thereof is written as a script into the error message (step 368). Setting is made such that the script which has been transmitted from the user is transmitted to the user (step 369). The script which has been transmitted from the user is returned to the user.

If no script error occurs (NO at step 367), it is judged whether or not a file representing an image composing an edit image defined by an edited script is a new file which is not stored in the hard disk for the server 1 (step 370).

If there is no new file (NO at step 370), the program proceeds to customize content reflection processing shown in FIG. 38.

IX. Customize Content Reflection Processing

Data representing an image selected by the user is updated in accordance with the script which has been transmitted from the user. The updated data is stored in the hard disk for the server 1 (step 392). Further, user information relating to a customized image is updated (step 393). Thereafter, the program proceeds to the customize menu processing shown in FIG. 33.

If there is a new file (YES at step 370), the program proceeds to new file transmission processing shown in FIG. 37.

X. New File Transmission Processing

In the server 1, data representing a file transmission image which is suitable for the user device and the browser is read

18

out (step 381). An error message is initialized, and a transmission file list is set in all new files (set such that all the new files are transmitted to the server) (step 382). Data representing the file transmission image, the error message, and the transmission file list are transmitted to the user device from the server 1 (step 383).

When the data representing the file transmission image, the error message, and the transmission file list are received in the user device (step 141), the received data are transmitted to the user display device (step 142). The file transmission image is displayed, as shown in FIG. 54, on the user device (step 142). A file name representing a new image file is entered by the user (step 143). The entered file name is described on the file transmission image. A file specified by the entered file name is read out of the user device, and is transmitted to the server 1 (step 144).

In the server 1, the file which has been transmitted from the user device is received in the server 1 (step 384). It is checked whether or not a script editing stop request is issued from the user (step 385) (the script editing stop request can be also set by the user, although the illustration thereof is omitted in FIG. 54).

If the script editing stop request is issued (YES at step 385), the program proceeds to the image editing main processing shown in FIG. 35.

If no script editing stop request is issued (NO at step 385), it is checked whether or not an error occurs in the transmitted file (steps 386 and 387).

If an error is included (YES at step 387), a file transfer error is written into an error message (step 388), the transmission file list is taken as an error file and is so set as to transmit a new file in which an error occurred again (step 389). Thereafter, the processing at the step 383 and the subsequent steps is repeated, so that the error message is transmitted to the user device from the server 1.

If no error is included (NO at step 387), the program proceeds to the customize content reflection processing shown in FIG. 38.

In the customize content reflection processing, the new file which has been received in the server 1 is stored in the server (step 391). Thereafter, data representing an image obtained by customizing the selected image is stored in the hard disk for the server 1 (step 392). Further, the user information is updated (step 393). Thereafter, the customize menu processing shown in FIG. 33 is repeated.

Returning to FIG. 21, when the user sets the category retrieval image in the menu image (YES at step 218), the program proceeds to category retrieval image processing shown in FIG. 26.

XI. Category Retrieval Image Processing

It is first confirmed by referring to the user information whether or not data representing an inherent category retrieval image which has been customized by the user (a user image) is stored in the hard disk for the server 1 (step 251). When there is data representing the category retrieval image inherent in the user (YES at step 252), the data representing the category retrieval image inherent in the user is read out of the hard disk (step 253). If the data representing the category retrieval image inherent in the user is not stored in the server 1 (NO at step 252), the data representing the category retrieval image which is suitable for the user device and the browser is read out of the hard disk for the server 1 (step 255). The data representing the category retrieval image which has been read out is transmitted to the user device from the server 1 (step 254).

When the data representing the category retrieval image which has been transmitted from the server 1 is received in the user device (step 81), the received data is fed to the user display device (step 82). On the user display device, a category retrieval image shown in FIG. 45 is displayed. A category to be retrieved is displayed on the category retrieval image. In FIG. 45, examples include "1. By artist", "2. By theme", "3. By year", "4. By country", and "5. By data format". Any one of the categories is selected by the user (step 83). Data representing the selected category is transmitted to the server 1 from the user device (step 84).

On the basis of data representing the category which has been received in the server 1 (step 256), work image retrieval processing is performed (step 257). Thereafter, the program proceeds to category retrieval result processing shown in FIG. 27.

XII. Category Retrieval Result Processing

Referring to FIG. 27, a category retrieval result image customized by the user is retrieved in the user information (step 261). If the user customizes the category retrieval result image, there is an image customized by the user (a user image). Accordingly, the category retrieval result image relating to the user is retrieved from the hard disk for the server 1 (step 263). If the user does not customize the category retrieval result image, a retrieval result image which is suitable for the user device and the browser out of default retrieval result images is read out of the hard disk for the server 1 (step 264). In either case, when the retrieval result image is read out of the hard disk for the server 1, the program proceeds to edit image production processing shown in FIG. 31.

VIII. Edit Image Production Processing

Referring to FIG. 31, an additional program suitable for an edit image to be produced is read out of the hard disk for the server 1 (step 301). In this case, an additional program suited to generate a retrieval result image (for example, a program for performing processing defined in a script by the user) is read out of the hard disk for the server 1. Files required to generate the retrieval result image (an image file, a sentence (text) file, a button file, a sound file, etc.) are read out of the hard disk for the server 1, thereby generating the result of retrieval (step 302). Data representing an edit image representing the produced result of retrieval is transmitted to the user device from the server 1 (step 303).

In the user device, the edit image data representing the result of retrieval which has been transmitted from the server 1 is received (step 91). The edit image data representing the result of retrieval is fed to the user display device, so that the edit image representing the result of retrieval is displayed on the user display device (step 92). A link destination (an image, a text, a URL name, etc.) which is displayed on the retrieval result image is designated by the user (step 93). Data representing the designated link designation is transmitted to the server 1 from the user device (step 94).

The data representing the link destination which has been transmitted from the user device is received in the server 1 (step 304). The program proceeds to the link processing shown in FIG. 32, so that data representing an image desired by the user will be transmitted to the user device from the server 1.

Returning to FIG. 21, when the user selects "3. Keyword Retrieval Image" in the menu image (see FIG. 43) displayed on the user display device, the program proceeds to keyword retrieval image processing shown in FIG. 24 (NO at step 218).

XIV. Keyword Retrieval Image Processing

Referring to FIG. 24, a keyword retrieval image customized by the user is retrieved on the basis of the user information (step 231). If there is a keyword retrieval image customized by the user (a user image) (YES at step 232), data representing the image is read out of the hard disk for the server 1 (step 233). If there is no keyword retrieval image customized by the user (NO at step 232), a default keyword retrieval image is read out of the hard disk for the server 1 (step 235). In either case, the data representing the keyword retrieval image which has been read out of the hard disk for the server 1 is transmitted to the user device from the server 1 (step 234).

In the user device, the data representing the keyword retrieval image which has been transmitted from the server 1 is received (step 75). The received data representing the keyword retrieval image is fed to the user display device, where a keyword retrieval image as shown in FIG. 44 is displayed (step 76).

An instruction "Enter keyword" is displayed on the keyword retrieval image. The user enters a keyword in accordance with the instruction (step 77). The entered keyword is displayed on the user display device. The entered keyword is transmitted to the server 1 from the user device (step 78).

In the server 1, data representing the keyword which has been transmitted from the user device is received (step 236). In the server 1, retrieval processing is performed on the basis of the received keyword (step 237). Consequently, the program proceeds to retrieval result image processing shown in FIG. 25.

XV. Retrieval Result Image Processing

In FIG. 25, it is judged by retrieval in the user information whether or not there is a retrieval result image customized by the user (step 241). When there is a retrieval result image customized by the user by the retrieval (YES at step 242), data representing the retrieval result image is read out of the hard disk for the server 1 (step 243). When there is no retrieval result image customized by the user (NO at step 242), data representing a predetermined retrieval result image which is suitable for the user device and the browser is read out of the hard disk for the server (step 244). Thereafter, the program proceeds to the edit image production processing shown in FIG. 31, so that a keyword retrieval result image is produced in the same manner as that in the above-mentioned category retrieval image production processing, and is transmitted to the user device.

The keyword retrieval result image is displayed on the user display device.

In the link processing (see FIG. 32), when a worklist image is displayed on the user display device (YES at step 318), the program proceeds to work list processing shown in FIG. 28.

XVI. Work List Processing

Referring to FIG. 28, a work list image which has been customized by the user is retrieved in the user information (step 271). If the work list image which has been customized by the user (a user image) exists in the server 1 (YES at step 272), data representing the work list image is read out of the hard disk for the server 1 (step 273). If no work list image which has been customized by the user exists in the server 1 (NO at step 272), data representing the work list image suitable for the user device out of predetermined work list images stored in the hard disk for the server 1 is read out

21

(step 274). Thereafter, the program proceeds to the edit image production processing shown in FIG. 31, so that the data representing the work list image is produced. The generated data representing the work list image is transmitted to the user device from the server 1. Accordingly, the work list image is displayed on the user display device.

FIGS. 46 to 49 respectively illustrate examples of the work list image.

FIG. 46 illustrates a work list image displayed when the user device is a portable telephone set (or a device capable of displaying only a text). In the portable telephone set, it is relatively difficult to display an image. Therefore, a work list image suitable for a display device in the portable telephone set is displayed.

FIG. 47 illustrates a work list image displayed when the user device is a personal digital assistant. Although the personal digital assistant can display an image, a display screen of a display device in the personal digital assistant is not so large. Therefore, a work list image which is not very large is displayed on the display device in the personal digital assistant.

FIG. 48 illustrates a work list image displayed when the user device is a personal computer, and a small display device is connected to the personal computer. A relatively large work list image is displayed.

FIG. 49 illustrates a work list image displayed when the user device is a personal computer, and a display device having a large display screen is connected thereto. A work list image suited to be displayed on the large display screen is displayed.

A small image is used in the work list image shown in FIG. 47, a medium-sized image is used in the work list image shown in FIG. 48, and a large image is used in the work list image shown in FIG. 49. Data representing the work list image which is composed of an image which differs in size depending on the user device is thus transmitted to the user device from the server 1. Color reduction processing will be also performed, as required.

XVII. Artist Image Processing

FIG. 29 illustrates artist image processing.

When a link with an artist image is established, an artist image which has been customized by the user is retrieved in the hard disk for the server 1 (step 281). If there is an artist image which has been customized by the user (YES at step 282), data representing the customized artist image is read out of the hard disk for the server 1 (step 283). If there is no artist image which has been customized by the user (NO at step 282), data representing the artist image which is suitable for the user device and the browser is read out of the hard disk for the server 1 (step 284). The data representing the artist image is generated from the data read out in the edit image production processing shown in FIG. 31. The generated data representing the artist image is transmitted to the user device, so that the artist image which is suitable for the user device and the browser or the artist image which has been customized by the user is displayed on the user display device.

XVIII. Work Image Processing

FIG. 30 shows work image processing.

When a link with a work image is established, a work image which has been customized by the user is retrieved in the hard disk for the server 1 (step 291). If there is a work image which has been customized by the user (a user image)

22

(YES at step 292), data representing the customized work image is read out of the hard disk for the server 1 (step 293). If there is no work image which has been customized by the user (NO at step 292), a work image which is suitable for the user device and the browser out of default work images is read out of the hard disk for the server 1 (step 294). In either case, data representing the work image is generated on the basis of processing relating to the work image which has been read out of the hard disk for the server 1, and is transmitted to the user device. The work image which has been customized by the user or the work image which is suitable for the user device and the browser is displayed on the user display device.

It goes without saying that when a work image, an artist image, or the like is displayed, a group of files is searched, as shown in FIGS. 8A to 10A and 10B, as described above, so that a corresponding file is used.

Returning to FIG. 35, if script editing is not performed, an image item is added, thereby customizing an image (NO at step 355).

A script is first initialized (step 356). Thereafter, the program proceeds to image item addition processing (step 357). The image item addition processing will be next described. When the image item addition processing is terminated, it is checked whether or not a new file is added in the image item addition processing (step 358). If the new file is added (YES at step 358), the program proceeds to the new file transmission processing shown in FIG. 37. Unless the new file is added (NO at step 358), the program proceeds to the customize content reflection processing shown in FIG. 38. In either case, the user information is updated by adding a display item by customization, as described above.

XIX. Image Item Addition Processing

FIG. 39 is a flow chart showing image item addition processing.

Data representing an addition item selection image which is suitable for the user device and the browser is read out of the hard disk for the server 1 (step 401). The data representing the addition item selection image which has been read out is transmitted to the user device from the server 1 (step 402).

In the user device, the data representing the addition item selection image is received (step 151), and is fed to the user display device (step 152). An addition item selection image as shown in FIG. 55 is displayed on the user display device (step 152).

A question text "Do you add layout or do you arrange file?" is displayed on the addition item selection image. The user selects desired processing out of "layout addition" and "file arrangement" depending on the text (step 153). Data representing the selected content is transmitted to the server 1 from the user device (step 154).

The data representing the selected content which has been transmitted from the user device is received in the server 1 (step 403). If the received content is a request to stop image editing (YES at step 404) (the stop of image editing and the termination of image editing, described later, can be set by the user, although the illustration thereof is omitted in FIG. 55), the program proceeds to the image editing main processing shown in FIG. 35, so that image editing processing is performed again, as required. If the received content is a request to terminate image editing (YES at step 405), the image item addition processing is terminated. If the received content is layout addition processing (YES at step 406), the program proceeds to layout addition processing shown in

23

FIG. 40 (step 407). If the received content is file arrangement processing (No at step 406), the program proceeds to file arrangement processing shown in FIG. 41 (NO at step 408).

XX. Layout Addition Processing

Referring to FIG. 40, description is made of the layout addition processing.

Data representing a layout selection image which is suitable for the user device and the browser is read out of the hard disk for the server 1 (step 411). The data read out is transmitted to the user device from the server 1 (step 412).

The data representing the layout selection image which has been transmitted from the server 1 is received in the user device (step 161). The data representing the layout selection image is fed to the user display device, so that a layout selection image as shown in FIG. 56 is displayed (step 162). An instruction "Enter layout to be changed" is displayed on the layout selection image. The user enters a layout to be changed, for example, right justification, left justification, centering, and multiple column of a text in accordance with the instruction (step 163). The entered layout instruction is transmitted to the server 1 from the user device (step 164).

The layout instruction which has been transmitted from the user device is received in the server 1 (step 413). If the received instruction is a request to stop image editing, the program proceeds to the image editing main processing shown in FIG. 35 (YES at step 414). The received instruction is a request to stop layout selection (YES at step 415), the layout addition processing is terminated (the stop of image editing or the stop of layout selection can be set by the user, although the instruction thereof is omitted in FIG. 56). Unless layout selection stop processing is performed (NO at step 415), a layout tag indicated by the user is added to a script for an image which is an object of the layout addition processing (step 416). Thereafter, in the above-mentioned image item addition processing, elements (an image, a text, a button, etc.) the layout of which is designated by the added tag are added to the script (step 417).

XXI. File Arrangement Processing

FIG. 41 is a flow chart showing file arrangement processing.

Data representing a file arrangement image which is suitable for the user device and the browser is read out of the hard disk for the server 1 (step 421). The data read out is transmitted to the user device from the server 1 (step 422).

In the user device, data representing the file arrangement image which has been transmitted from the server 1 is received (step 171). A file arrangement image as shown in FIG. 57 is displayed on the user display device.

The file arrangement image includes the following areas:

Arrangement file area 501;

A arrangement file area 501 includes an area 502 checked by the user when a title text file is arranged, an area 503 checked by the user when an artist image file is arranged, and an area 504 checked by the user when a work image file is arranged. The areas 502, 503, and 504 corresponding to a file which should be arranged by the user are checked by the user.

Display object area 505;

A display object area 505 is for designating an object (for example, an image, an artist name, a title, etc.) to be displayed in the file (management file) selected in the arrangement file area 501.

24

Display option area 510;

A display option area 510 further includes the following areas:

Link setting area 511;

It is an area for setting a link destination.

Retrieval condition area 512;

It is for designating a retrieval object (condition) to be hit by retrieval. For example, when only works issued after a certain year are taken as retrieval objects, the year is entered by the user.

Display areas 513 and 514;

They are for designating an area where an image, a text, or a button is to be displayed. The width is entered in the area 513, and the length is entered in the area 514.

Color designation area 515;

It is an area for entering a color number for specifying a color to be displayed.

Number-of-colors designation area 516;

It is an area for designating the number of colors to be displayed. For example, 256 colors, 16,000,000 colors, or the like is entered.

Font designation area 517;

It is an area for designating a font to be displayed. A Gothic type or a Mincho type, for example, is entered.

Point designation area 518;

It is an area for designating the size of characters.

Enhancement designation area 519;

It is an area used when characters are enhanced. For example, an underline, reverse characters, hollow characters, or the like is entered.

In addition thereto, the file arrangement image includes an area 521 clicked by the user when arrangement is determined, an arrangement stop area 522 clicked by the user when arrangement processing is stopped, and an editing stop area 523 clicked by the user when editing is stopped.

Predetermined selection processing is performed using the file selection image by the user (step 173). Data representing the selected content is transmitted to the server 1 from the user device (step 174).

The data representing the selected content which has been transmitted from the user is received in the server 1 (step 423). If the received content is a request to stop image editing (YES at step 424), the program proceeds to the image editing main processing shown in FIG. 35. If file arrangement stop processing is performed (YES at step 425), the file arrangement processing is terminated. Unless the file arrangement stop processing is performed, a file tag is added to a script for an image to be an editing object depending on processing selected by the user (step 426). Consequently, a file is arranged at a position selected by the user.

The user can customize an image displayed on the user display device relatively freely. Moreover, when the image is not customized, an image which is suitable for the user device and the browser is displayed on the user display device.

Although the present invention has been described and illustrated in detail, it is clearly understood that the same is by way of illustration and example only and is not to be taken by way of limitation, the spirit and scope of the present invention being limited only by the terms of the appended claims.

What is claimed:

1. An image communication system comprising an image transmitting apparatus and an image receiving apparatus which can establish data communication with each other, wherein

25

said image receiving apparatus comprises
 a device information transmission device for transmitting to said image transmitting apparatus information relating to said image receiving apparatus, and
 said image transmitting apparatus comprises
 a device information receiving device for receiving information relating to said image receiving apparatus which has been transmitted from said device information transmission device in said image receiving apparatus, and
 an image data transmission device for transmitting to said image receiving apparatus image data representing an image having a display format suitable for said image receiving apparatus on the basis of the information relating to the image receiving apparatus which has been received by said device information receiving device,
 wherein said received information comprises information about a type of browser installed in said image receiving apparatus and at least one of a number of colors displayable on a display of said image receiving apparatus, and a size of a displayable image.

2. The image a communication system according to claim 1, wherein
 said device information transmission device in said image receiving apparatus transmits to said image transmitting apparatus at least one of the type of said image receiving apparatus, information relating to display on the display device, and information relating to software for the display of an image which is installed in said image receiving apparatus.

3. The image communication system according to claim 2, wherein said type comprises at least one of a portable telephone, a portable information terminal, a notebook computer, and a personal computer, and
 wherein said information relating to display on the display device comprises at least one of a size of a display screen and a number of colors which can be displayed.

4. The image communication system according to claim 1, wherein
 said image transmitting apparatus further comprises a storage device for storing image data having a plurality of different display formats, and
 said image data transmission device transmits to said image receiving apparatus image data representing the image having the display format suitable for said image receiving apparatus from said storage device on the basis of the information relating to the image receiving apparatus which has been received by said device information receiving device.

5. The image communication system according to claim 1, wherein
 said image data transmission device in said image transmitting apparatus transmits to said image receiving apparatus edit image data representing an edit image having a plurality of display items including an image and a sentence.

6. The image communication system according to claim 1, wherein
 said image receiving apparatus further comprises
 a setting device for setting at least one of the layout and said display items of said edit image, and
 a set information transmission device for transmitting to said image transmitting apparatus information set by said setting device, and

26

said image transmitting apparatus comprises
 set information receiving device for receiving the set information which has been transmitted from said set information transmission device in said image receiving apparatus, and
 a determination device for determining at least one of the layout and the display items of the edit image represented by said edit image data on the basis of the set information which has been received by said set information receiving device.

7. The image communication system according to claim 6, wherein
 said setting device in said image receiving apparatus is at least one of means for performing the setting so as to collectively edit the whole of said edit image and means for performing the setting so as to edit a part of said edit image.

8. The image communication system according to claim 6, wherein
 the image data representing images corresponding to a plurality of frames are transmitted to said image receiving apparatus from said image transmitting apparatus, said image receiving apparatus further comprises an edit image designation device for designating said edit image having the layout and said display items at least one of which should be set by said setting device, and said setting device sets at least one of said layout and said display items of the edit image designated by said edit image designation device.

9. The image communication system according to claim 5, wherein said image transmitting apparatus further comprises:
 plural image data storage devices storing a plurality of image data representing images composing said edit image in different data amounts, and
 a production device for producing said edit image using the image represented by any one of the image data stored in said plural image data storage devices.

10. The image communication system according to claim 1, wherein
 said image receiving apparatus further comprises
 a selection device for selecting the image first displayed when said image receiving apparatus and said image transmitting apparatus are formally connected to each other, and
 a selected image transmission device for transmitting to said image transmitting apparatus selected image data representing the image selected by said selection device,
 said image transmitting apparatus further comprises
 a selected image receiving device for receiving the selected image data which has been transmitted from said selected image transmission device in said image receiving apparatus, and
 said image data transmission device transmits image data representing the image represented by the selected image data which has been received by said selected image receiving device to said image receiving apparatus when the image receiving apparatus and the image transmitting apparatus are formally connected to each other.

11. The image communication system according to claim 10, wherein
 said image receiving apparatus further comprises
 a link image setting device for setting said image to be linked, and

27

a link information transmission device for transmitting link information set by said link image settings device to said image transmitting apparatus, said image transmitting apparatus further comprises a link information receiving device for receiving the link information which has been transmitted from said link information transmission device in said image receiving apparatus, and said image data transmission device transmits to said receiving apparatus image data representing the subsequent image to be linked on the basis of the link information which has been received by said link information receiving device.

12. The image communication system according to claim 1, wherein

said image receiving apparatus further comprises an update command transmission device for transmitting an image update command to said image transmitting apparatus,

said image transmitting apparatus further comprises an update command receiving device for receiving said image update command which has been transmitted from said update command transmission device, and said image data transmission device transmits to said image receiving apparatus image data representing the subsequent image which is linked to an image represented by image data which corresponds to the update command received by said update command receiving device and information relating to said image receiving apparatus and has been transmitted to said image receiving apparatus by said image data transmission device.

13. The image communication system according to claim 1, wherein said image received by said image receiving apparatus is edited by a user.

14. The image communication system according to claim 1, wherein said image receiving apparatus displays one of said image and an image customized by a user.

15. The image communication system according to claim 1, wherein said image transmitting apparatus stores user preferences for a display format suitable for said image receiving apparatus.

16. The image communication system according to claim 1, wherein said image transmitting apparatus stores user information, said user information including at least one of a user name and password, a script for a start image, retrieval conditions, a script for a menu image, a script for a category retrieval image, a script for a work list image, a script for an artist image, and a script for a work image.

17. The image communication system according to claim 1, wherein an initial image to be displayed is selected by a user of said image receiving device and information related to the user selection is stored by said image transmitting device.

18. An image transmitting apparatus which can establish data communication with an image receiving apparatus, comprising:

a device information receiving device for receiving information relating to the image receiving apparatus which has been transmitted from said image receiving apparatus; and

an image data transmission device for transmitting to said image receiving apparatus image data representing an image having a display format suitable for said image receiving apparatus on the basis of the information relating to the image receiving apparatus which has been received by said device information receiving device,

28

wherein said received information comprises information about a type of browser installed in said image receiving apparatus and at least one of a number of colors displayable on a display of said image receiving apparatus, and a size of a displayable image.

19. In an image transmitting apparatus which can establish data communication with an image receiving apparatus, an image data transmitting method comprising:

receiving information relating to the image receiving apparatus which has been transmitted from said image receiving apparatus; and

transmitting to said image receiving apparatus image data representing an image having a display format suitable for said image receiving apparatus on the basis of said received information relating to the image receiving apparatus,

wherein said received information comprises information about a type of browser installed in said image receiving apparatus and at least one of a number of colors displayable on a display of said image receiving apparatus, and a size of a displayable image.

20. A computer readable recording medium storing a program for controlling a computer in an image transmitting apparatus which can establish data communication with an image receiving apparatus so as to:

receive information relating to the image receiving apparatus which has been transmitted from said image receiving apparatus, and

transmit to said image receiving apparatus image data representing an image having a display format suitable for said image receiving apparatus on the basis of the received information relating to the image receiving apparatus,

wherein said received information comprises information about a type of browser installed in said image receiving apparatus and at least one of a number of colors displayable on a display of said image receiving apparatus, and a size of a displayable image.

21. An image communication system comprising an image transmitting device and an image receiving device which can establish data communication with each other, wherein

said image receiving device comprises

device information transmission means for transmitting to said image transmitting device information relating to said image receiving device, and

said image transmitting device comprises

device information receiving means for receiving information related to said image receiving device which has been transmitted from said device information transmission means in said image receiving device, and

image data transmission means for transmitting to said image receiving device image data representing an image having a display format suitable for said image receiving device on the basis of the information relating to the image receiving device which has been received by said device information receiving means,

wherein said received information comprises information about a type of browser installed in said image receiving means and at least one of a number of colors displayable on a display of said image receiving means and a size of a displayable image.

22. An image transmitting device which can establish data communication with an image receiving device, comprising:

29

device information receiving means for receiving information relating to the image receiving device which has been transmitting from said image receiving device; and

image data transmission means for transmitting to said image receiving device image data representing an image having a display format suitable for said image receiving device on the basis of the information relating to the image receiving device which has been received by said device information receiving means,

wherein said received information comprises information about a type of browser installed in said image receiving means and at least one of a number of colors displayable on a display of said image receiving means, and a size of a displayable image.

23. An image communication system comprising:

an image transmitting apparatus; and

an image receiving apparatus which can establish data communication with each other,

wherein said image receiving apparatus comprises:

a device information transmission device for transmitting to said image transmitting apparatus information relating to said image receiving apparatus, and

wherein said image transmitting apparatus comprises:

a device information receiving device for receiving information relating to said image receiving apparatus which has been transmitted from said device information transmission device in said image receiving apparatus; and

an image data transmission device for transmitting to said image receiving apparatus image data representing an image having a display format suitable for said image receiving apparatus on the basis of the information relating to the image receiving apparatus which has been received by said device information receiving device,

wherein said image transmitting device stores a plurality of initial images to be displayed, said plurality of initial images being selected by a plurality of users.

24. The image communication system according to claim 23, wherein each one of said plurality of initial images selected by said plurality of users comprises a unique image.

25. The image communication system according to claim 23, wherein said image transmitting apparatus comprises a server, and

wherein said image receiving apparatus comprises at least one of a portable telephone, a portable information terminal, a notebook computer, and a personal computer.

26. The image communication system according to claim 25, wherein the image received by said at least one of a portable telephone, a portable information terminal, a notebook computer, and a personal computer is editable by a user.

27. The image communication system according to claim 23, wherein said image transmitting device stores a link destination of an image edited by a user.

28. The image communication system according to claim 27, wherein the link destination of an image selected by a user is transmitted from the image receiving device to the image transmitting device.

30

29. The image communication system according to claim 27, wherein the link destination of an image selected by each user of a plurality of users is unique.

30. An image communication system comprising:

an image transmitting apparatus; and

an image receiving apparatus which can establish data communication with each other,

wherein said image receiving apparatus comprises:

a device information transmission device for transmitting to said image transmitting apparatus information relating to said image receiving apparatus, and

wherein said image transmitting apparatus comprises:

a device information receiving device for receiving information relating to said image receiving apparatus which has been transmitted from said device information transmission device in said image receiving apparatus; and

an image data transmission device for transmitting to said image receiving apparatus image data representing an image having a display format suitable for said image receiving apparatus on the basis of the information relating to the image receiving apparatus which has been received by said device information receiving device,

wherein said image transmitting apparatus comprises a server, and

wherein said image receiving apparatus comprises at least one of a portable telephone, a portable information terminal, a notebook computer, and a personal computer.

31. An image communication system comprising:

an image transmitting apparatus; and

an image receiving apparatus which can establish data communication with each other,

wherein said image receiving apparatus comprises:

a device information transmission device for transmitting to said image transmitting apparatus information relating to said image receiving apparatus, and

wherein said image transmitting apparatus comprises:

a device information receiving device for receiving information relating to said image receiving apparatus which has been transmitted from said device information transmission device in said image receiving apparatus, and

an image data transmission device for transmitting to said image receiving apparatus image data representing an image having a display format suitable for said image receiving apparatus on the basis of the information relating to the image receiving apparatus which has been received by said device information receiving device,

wherein said image transmitting device stores a plurality of initial images to be displayed, said plurality of initial images selected by a plurality of users,

wherein a link destination for each of said plurality of initial images is transmitted from the image receiving device to the image transmitting device, and

wherein said image transmitting device stores said link destination for each of said plurality of initial images to be displayed.

* * * * *



US006691281B1

(12) **United States Patent**
Sorge et al.

(10) Patent No.: **US 6,691,281 B1**
(45) Date of Patent: **Feb. 10, 2004**

(54) **PUBLISHING/REPUBLISHING DATA TABLES IN HTML DOCUMENTS WHILE MAINTAINING FORMATTING AND FUNCTIONALITY FOR RESTORING BACK THE DATA TABLES**

6,182,092 B1 * 1/2001 Francis et al. 707/513
6,230,173 B1 * 5/2001 Ferrel et al. 707/501.1
6,396,500 B1 * 5/2002 Qureshi et al. 345/473

OTHER PUBLICATIONS

Moseley et al., "Microsoft Office 97", copyright 1997 SYBEX Inc., pp. 531, 1031-1041 in view of Francis et al., US 6,182,092 B1 filed Jul. 1997.*

Microsoft Corporation. *Getting Results with Microsoft® Office 97*. "Publish Microsoft Excel Tables and Charts on the Web." ©1995-1997 Microsoft Corporation. pp. 448-451.

* cited by examiner

Primary Examiner—Stephen S. Hong

Assistant Examiner—Thu V. Huynh

(74) Attorney, Agent, or Firm—Ronald M. Anderson

(57) ABSTRACT

A spreadsheet program directly publishes a data table or chart into an HTML document. The table or chart is published to a predefined location within the HTML document and a user is enabled to readily update the published table or chart to include changes made in the spreadsheet program by republishing the data table or chart. Furthermore, the published or republished data can be imported back into the parent spreadsheet program from the HTML document without loss of functionality or formatting that it had in the parent spreadsheet program. In addition, the data may be published in a form that can be used by ActiveX web components in order to provide spreadsheet functionality from within the browser application. A unique marker tag or identification tag within the HTML document indicates where the data table or chart has been inserted.

(75) Inventors: Terri L. Sorge, Kirkland, WA (US);
May May Quan, Kirkland, WA (US);
Kent R. Lowry, Seattle, WA (US);
Russell S. Johnson, Seattle, WA (US);
John L. Dauphiny, Woodinville, WA (US)

(73) Assignee: Microsoft Corporation, Redmond, WA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: 09/333,816

(22) Filed: Jun. 15, 1999

(51) Int. Cl.⁷ G06F 15/00

(52) U.S. Cl. 715/503; 715/509; 715/513;
715/523

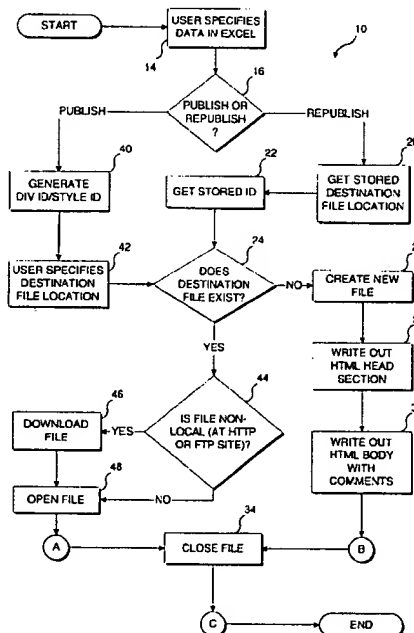
(58) Field of Search 707/504, 503,
707/516, 523, 515; 715/503, 509, 513,
516, 523

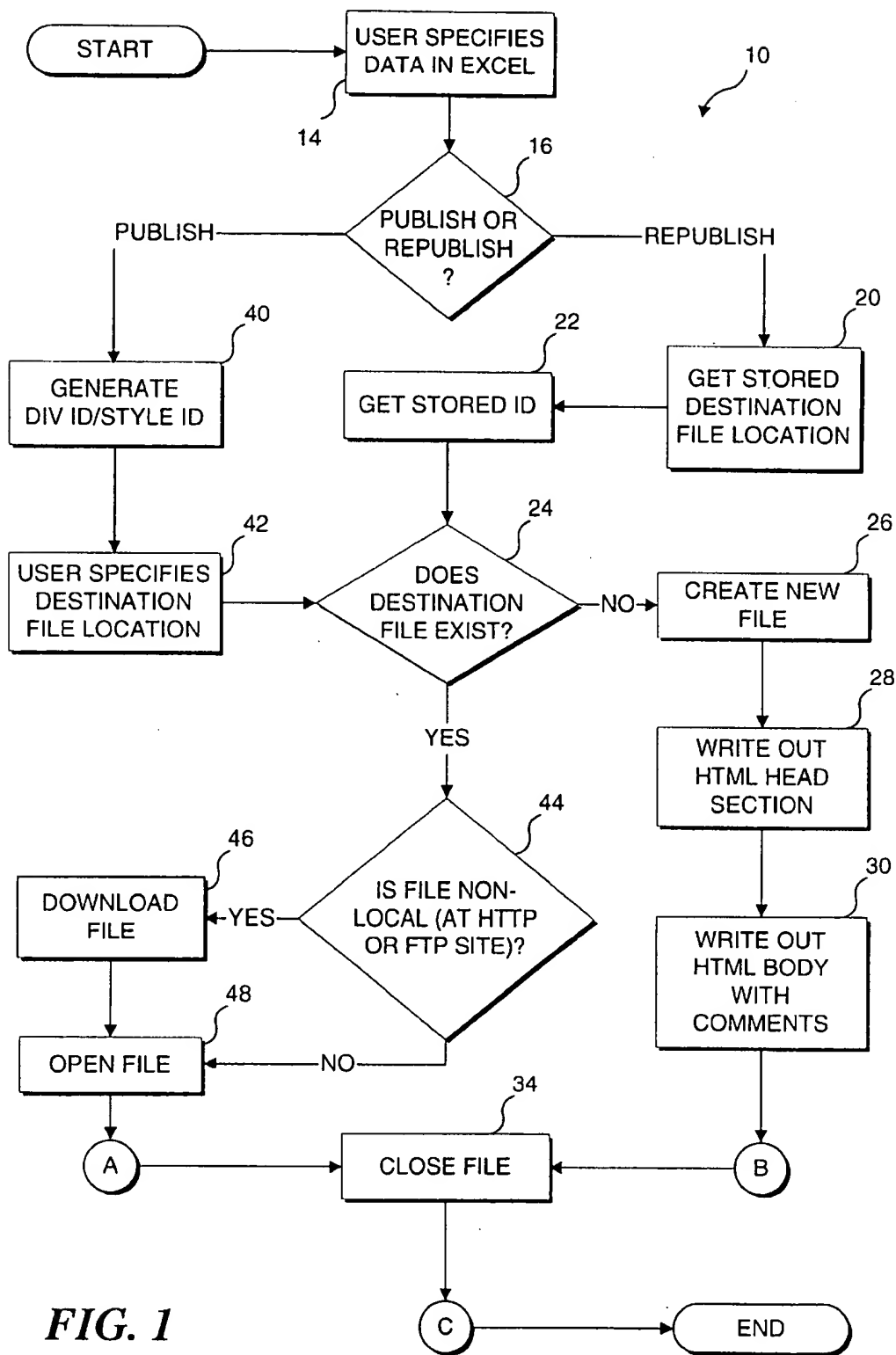
(56) References Cited

U.S. PATENT DOCUMENTS

5,860,073 A * 1/1999 Ferrel et al. 707/513
5,991,760 A * 11/1999 Gauvin et al. 707/10
6,078,924 A * 6/2000 Ainsbury et al. 707/101

28 Claims, 5 Drawing Sheets



**FIG. 1**

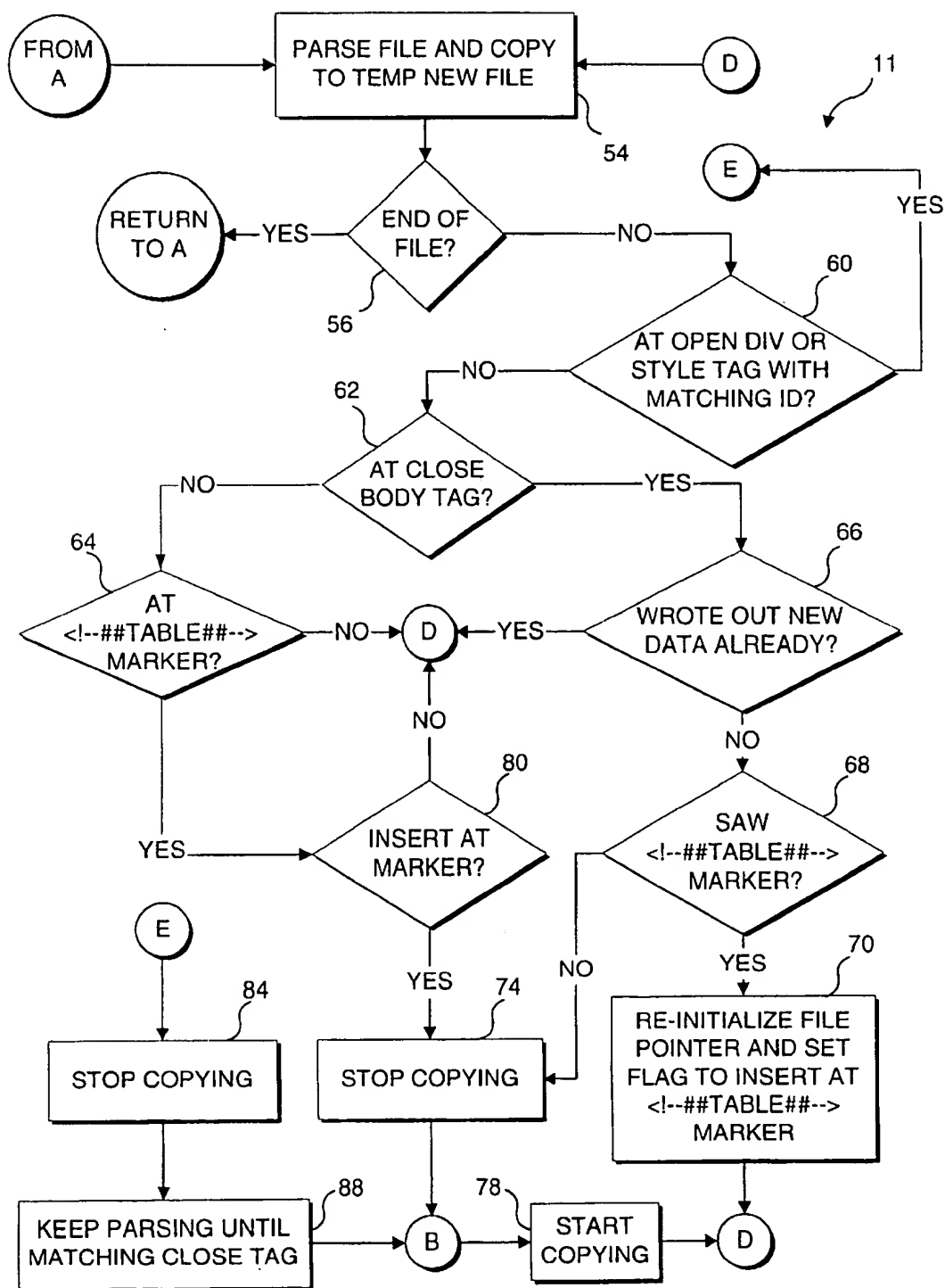
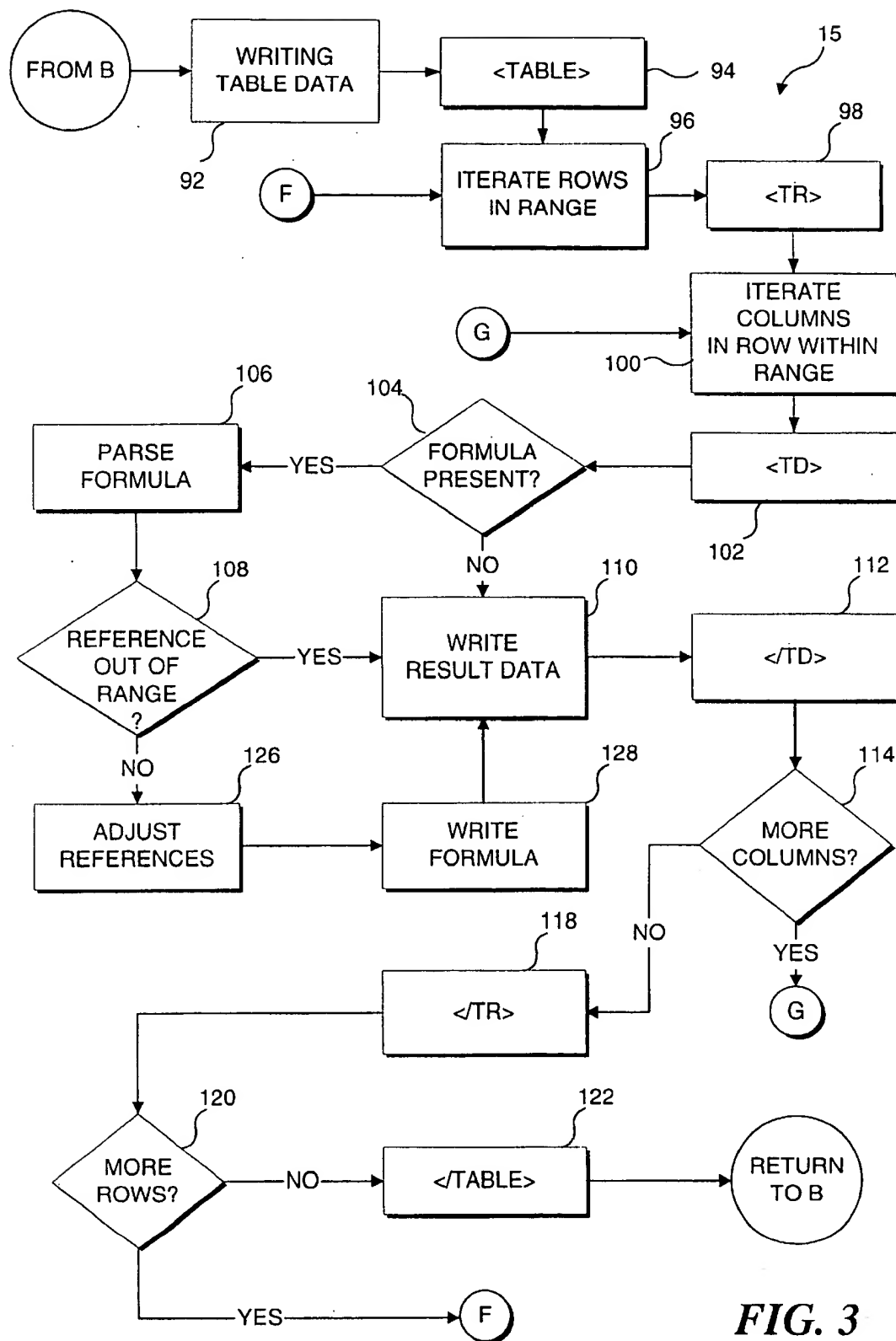
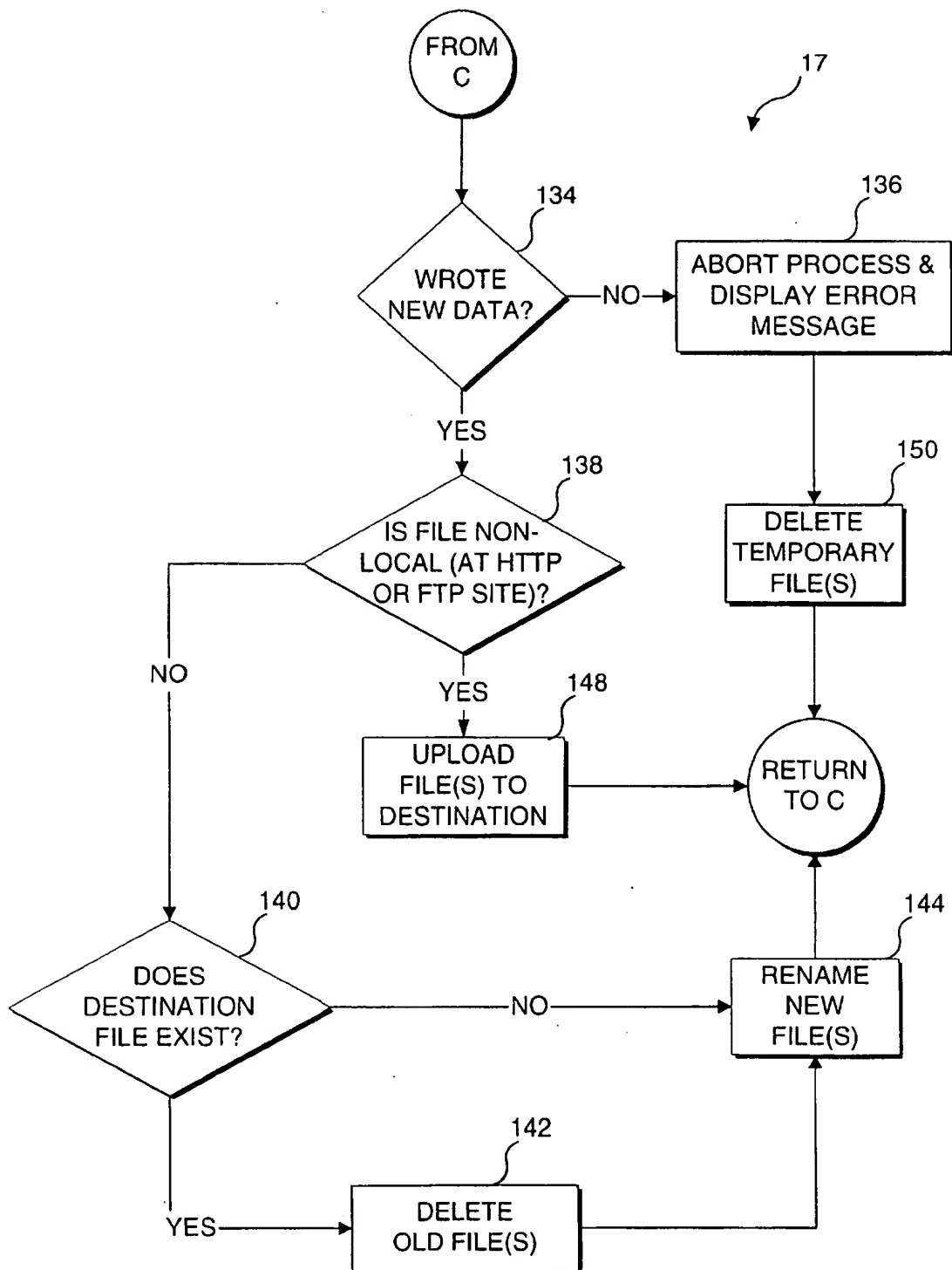


FIG. 2

**FIG. 3**

**FIG. 4**

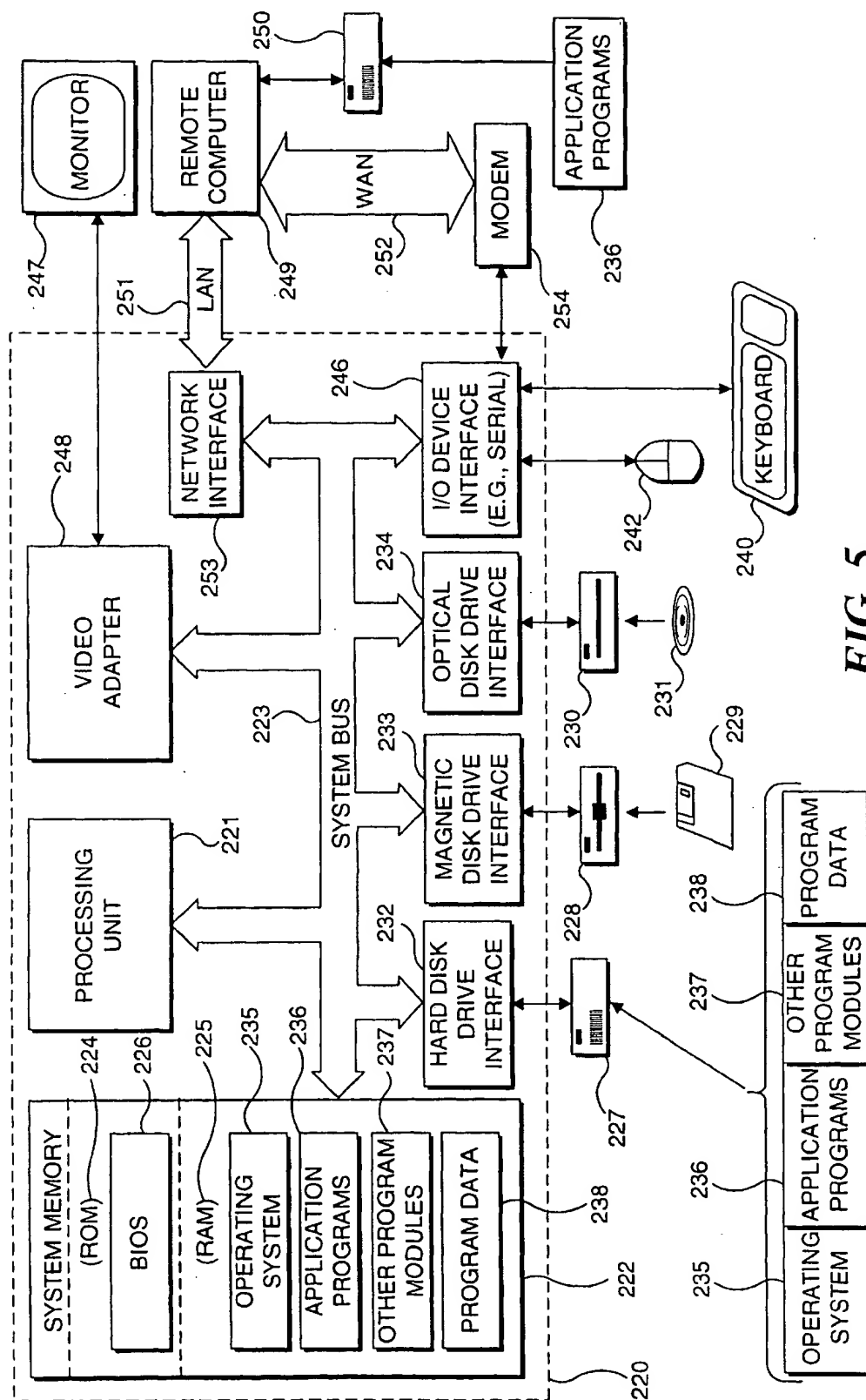


FIG. 5

1

PUBLISHING/REPUBLISHING DATA TABLES IN HTML DOCUMENTS WHILE MAINTAINING FORMATTING AND FUNCTIONALITY FOR RESTORING BACK THE DATA TABLES

FIELD OF THE INVENTION

The present invention generally pertains to exporting data into hypertext markup language (HTML) documents, and more specifically, to updating a previously exported discreet data section in an HTML document, without changing any other section of the HTML document; and while ensuring that such data can be imported from the HTML document back into a source application with all the formatting unique to the source application intact.

BACKGROUND OF THE INVENTION

With the increasing popularity of the Internet, HTML documents or files has become the internationally accepted format for sharing data "on-line." An on-line information system typically includes a server computer system that makes information available so that client computer systems can access the information. The server and client computer systems are usually connected in either a local area or a wide area private Intranet system, or via the public Internet. A unique uniform resource locator (URL) is associated with each HTML document, enabling the client computer systems to request a specific HTML document from a server computer system.

An HTML document includes a hierarchical set of markup elements; most elements have a start tag, followed by content, followed by an end tag. The content is typically a combination of text and nested markup elements. Tags, which are enclosed in angle brackets ('<' and '>'), indicate how the document is structured and how to display the document, i.e., its format. There are tags for markup elements such as titles and headers, for text attributes such as bold and italic, for lists, for paragraph boundaries, for links to other documents or other parts of the same document, for graphic images, for non-displayed comments, and for many other features. Further details regarding HTML may be found in reference books such as, "HTML For Dummies," by Ed Tittel and Steve James (1996).

The following lines of HTML briefly illustrate how the language is used:

Here we start a new paragraph <P>.

Some words are bold, others are <I>italic</I>. The viewer of the document will see:

Here we start a new paragraph. \

Some words are bold, others are italic.

As noted above, a user who wishes to retrieve and display an HTML document generally uses a Web browser program. Two of the popular Web browser programs are: NAVIGATOR™ from Netscape Communications Corp. of Mountain View, Calif., and INTERNET EXPLORER™ from Microsoft Corporation of Redmond, Wash. The primary functionality of web browsers is directed to finding, retrieving, and displaying documents. A browser is generally not intended for word processing or data manipulation of the information contained within an HTML document, but can display documents or data generated by word processing or spreadsheet applications, once converted into an appropriate HTML compatible format.

A wide variety of data may be shared among different users in a network environment using HTML. Typical

2

HTML documents include images, text, and data. HTML documents can be created using programs specifically designed for that purpose, such as Microsoft Corporation's FRONTPAGE™ Web Page publishing program.

5 Additionally, some applications, such as Microsoft Corporation's WORD 97™ word processing program, allow a user to save a text document as an HTML document. Microsoft Corporation's EXCEL 97™ spreadsheet program also enables a user to export a data table or chart created on a worksheet into an existing or new HTML document. If EXCEL 97 is used to export a data table or chart into an existing HTML document, the user is required to use a separate HTML editor to insert a marker tag, <!--
10 ##Table##-->, into the HTML document at the location where the table is to be inserted. EXCEL 97 will then convert the spreadsheet data table into the HTML compatible format and insert the data at the location in the HTML document specified by the marker tag. It would be desirable to be able to export a spreadsheet data table converted into HTML compatible format into an HTML document without requiring that an HTML editing application insert a <!--
15 ##Table##--> marker tag.

A characteristic of data tables or charts incorporated into HTML documents using prior art methods is that once the data tables or charts are imported into the HTML document, they lose virtually all of the functionality that they had in the spreadsheet application. Furthermore, a data table published into an HTML document cannot be reintroduced into its parent application with its original functionality intact, because critical formatting information unique to the parent spreadsheet application, such as any formulas included in the data table, are not maintained. Thus, formulas and other spreadsheet application unique parameters are lost in the publication process. Manipulation of the data from the HTML document within the parent application is thus not available.

Often, it would be desirable to enable changes to data in a table or chart published into an HTML document. For example, assume that an HTML document created by a real estate brokerage firm includes a data table for determining monthly mortgage payments based on the amount of the loan (rounded to the nearest \$1000), a current annual interest rate, and a fixed term. Additionally, the web page may include some text relating to the real estate services offered by the brokerage firm and a few testimonials from satisfied customers. Prospective purchasers of real estate may use the table to determine the monthly payments on a property. The table is most easily created in a spreadsheet application, converted into an HTML compatible format, and inserted into an HTML document that includes text produced with an HTML editor.

After several weeks, the brokerage firm may recognize that the table is less useful than desired because the interest rate has changed. To modify the table so that it is based on the newer current interest rate, the broker must create a new table with the spreadsheet application, convert the new table into the HTML compatible format, and then insert the revised table into the HTML document. Note that using only an HTML editor to change the original imported table would require manual input of not just the new interest rate, but also each and every monthly payment listed in the table. The data table in the HTML document does not have any spreadsheet functionality, and changing the interest rate value used in the spreadsheet formula to calculate the monthly payments does not result in an automatic recalculation of the monthly payments values shown in the table within the HTML document. Clearly, it would be desirable

3

to enable a chart or data table in an existing HTML document to be updated using the spreadsheet application, rather than by use of an HTML editor.

It would also be desirable to avoid the need to recreate the data table in the spreadsheet application simply to make a change. While recreating the data table is not necessary if the original data table was saved in its spreadsheet format, there will be times when the spreadsheet format of the data table has not been saved or has been deleted. Under the prior art, if an attempt is made to import the data table or chart from the HTML document back into the spreadsheet application for revisions, any formulas or formatting unique to the spreadsheet application originally used to create that data table or chart would no longer be present, having been lost in the process of translating the data table or chart into the HTML compatible format. Thus, the prior art cannot successfully "roundtrip" the data represented by the data table or chart back from the spreadsheet application, into an HTML document, and then back into the spreadsheet application. It would be desirable for data tables or charts created in a spreadsheet application to be readily inserted into an HTML document, and then subsequently imported back into the spreadsheet application without loss of functionality.

Currently, a data table or chart loses its computational functionality once it is exported from a spreadsheet application into an HTML document. The data table simply becomes a table of entries in an HTML document. Any change in an entry is incapable of causing a change in any other entry. Certainly, it would be more useful if the imported data table or chart retained its spreadsheet functionality, so that a change in one of the cells would cause any other values that depend upon the changed cell to be recalculated. In the previous example of an HTML document for a real estate brokerage firm, an active data table that retains its spreadsheet functionality would enable a user to enter a loan amount, the term, and the current interest rate, causing the monthly mortgage payment based upon these parameters to be determined and displayed.

SUMMARY OF THE INVENTION

In accord with the present invention, a method is defined for using an application program in which data having a format and a functionality specific to the application program are created, to publish at least a portion of the data into an HTML document such that when changes are made to the original data selection in the application program, those changes can be readily republished into the HTML document to replace the previously published data selection. Furthermore, when the HTML document containing the published or republished data selection is subsequently opened in the application program, the data selection retains the formatting and functionality that it originally had in the application program. The method uses the application program to select the portion of the data to be published. The data selected are then translated into an HTML compatible format, while preserving parameters that define the format and functionality of the data within the application program used to create the data. Once translated, the translated data are inserted into the HTML document.

Preferably, the application program is a spreadsheet program and the data that are translated comprise either a data table or a chart.

This method preferably includes the step of creating a unique identifier (ID) that is associated with the data that are translated and inserted into the HTML document. The unique ID is stored with the original data from which the portion of the data is selected and also included within the

4

HTML document. The translated data can then replace an older version of the data that were translated and previously inserted into the HTML document by the application program, thereby republishing the data to include changes made after the data were previously published. Reference to the unique ID enables the application program to identify the older version of the data and replace the older version of the data within the HTML document with the translated data comprising the newer version.

Consider the case where the application program is a spreadsheet program and the data selection is a data table that includes a plurality of cells. If at least one cell includes a formula that returns a calculated value in response to a change in a parameter in another of the plurality of cells, the translated data table inserted into the HTML document retains the functionality of the formula, so that a change in the parameter within the HTML document causes a recalculation of the formula to return a new calculated value that is retained when the HTML document is loaded back into the application program.

Preferably, the step of preserving the format of the translated data comprises the step of storing formatting information for the translated data within a style section in the HTML document.

The application program preferably inserts tags into the HTML document, indicating a beginning and end of the translated data inserted, and indicating functional elements of the data.

Other aspects of the present invention are directed to an article of manufacture that includes a memory media for carrying out functions generally consistent with the steps of the method described above, and to a system that includes a memory for storing machine instructions and a processor. When the machine instructions are executed by the processor, they implement functions that are also generally consistent with the steps of the above-described method.

BRIEF DESCRIPTION OF THE DRAWING FIGURES

The foregoing aspects and many of the attendant advantages of this invention will become more readily appreciated as the same becomes better understood by reference to the following detailed description, when taken in conjunction with the accompanying drawings, wherein:

FIG. 1 is a flow chart illustrating the logical steps implemented to publish or republish a data table into an HTML document from a source application, in accord with the present invention;

FIG. 2 is a flow chart illustrating the logical steps implemented to complete a parsing routine used when publishing or republishing a data table into an HTML document from a source application, in accord with the present invention;

FIG. 3 is a flow chart illustrating the logical steps implemented to complete a data table writing subroutine, in accord with the present invention;

FIG. 4 is a flow chart illustrating the logical steps implemented to complete a close file subroutine when publishing or republishing a data table into an HTML document from a source application, in accord with the present invention;

FIG. 5 is a block diagram of a personal computer system suitable for implementing the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

A preferred embodiment of the present invention will be included in the EXCEL 2000™ spreadsheet program, which

5

will be distributed by Microsoft Corporation as part of its OFFICE 2000™ product line. As implemented therein, the present invention will enable a user or creator of a spreadsheet to publish a data table from the spreadsheet into any HTML document at a preselected location without requiring a separate HTML editor to define the location with a marker tag. In addition, the user can easily republish or refresh a data table that had been previously published by the spreadsheet program into an HTML document to reflect changes in the data table that were made in the spreadsheet application. Furthermore, the present invention enables a user to reintroduce a data table previously published into an HTML document back into the spreadsheet application without loss of formatting or functionality. Before explaining how these features are implemented, it will be helpful to define several terms.

The term "publish" as used herein and in the claims that follow means to export data (such as a data table or a chart) from a source application (such as a spreadsheet program or a database program) into an HTML document. The terms "republish" and "refresh" as used herein and in the claims that follow mean updating that data that was previously published into an HTML document, to include any changes subsequently made to the data, using the source application in which the data was originally created. The updating process can be limited to refreshing data values or can be a more detailed republishing in which significant revisions to the data are made. The term "HTML document" as used herein and in the claims that follow means a file that includes HTML content, which is intended to be viewed or displayed with a Web browser. The term "data" as used herein and in the claims that follow means any information produced by a spreadsheet, database, or word processing application, such tables, charts, text, or images. The following example of a preferred embodiment of the present invention is applied to a spreadsheet application, generally as will be implemented in Microsoft Corporation's EXCEL 2000 spreadsheet application; however, it is not intended that the invention be limited to this application, since it can be clearly applied to other types of data, created in other types of applications. For simplicity, all further references to the EXCEL 2000 program in the following description will omit reference to the term "spreadsheet program."

FIG. 1 illustrates the logical steps used in EXCEL 2000 to publish or republish a data table into an HTML document. In a block 14, the user specifies or selects data in EXCEL 2000 to be published or republished into an HTML document, for example by selecting a range of contiguous cells using a cursor controlled by a mouse or other pointing device. A decision block 16 prompts the user to indicate whether the data selected are to be published or republished. If the user indicates that the selected data are to be published, the selected data will be inserted into an existing HTML document, or a new HTML document may be created that will include the EXCEL 2000 data. The republish/refresh feature enables an EXCEL 2000 user to update a data table or chart previously published from EXCEL 2000 into an HTML document.

For instance, an EXCEL 2000 user may have published a data table representing real estate mortgage payment calculations into an HTML document. Assume the data table lists a plurality of loan amounts in \$10,000 increments, the current interest rate for home loans, and calculates the required monthly payment based on each loan amount and the interest rate. A person viewing the HTML document in a Web browser would be able to select the loan amount most closely representing the loan required for a particular pur-

6

chase to determine the monthly mortgage payment. Because interest rates are subject to frequent change, it is likely that the creator of such a table would want to regularly update that table to reflect the current interest rate.

The EXCEL 2000 republish feature easily allows such a user to enter EXCEL 2000, open the appropriate data table that was previously saved as a spreadsheet, enter the current interest rate, and then republish the data table into the HTML document. Under the prior art, it would have been necessary to use an HTML editor application to edit not just the interest rate in the HTML document, but also all of the monthly payment results for each loan amount value that existed in the data table. Alternatively, an entirely new data table could be created and published, replacing the old table in the HTML document. When using EXCEL 2000 to republish the new version of the table, the user only needs to enter the new interest rate into EXCEL 2000, which recalculates the data table. The user then republishes the data table into the HTML document, changing all of the values for the monthly payment. The republication function is thus a much simpler process than any available in the prior art.

Assuming a user selects the publish feature at decision block 16, the logic proceeds to a block 40. In a block 40, EXCEL 2000 generates a unique ID attribute for the DIV tag. The ID attribute is an important feature that facilitates republishing of data in an HTML document. In order to make republishing efficient, EXCEL 2000 needs to be able to find the information to be replaced. This approach is particularly helpful when an HTML document contains multiple data tables published from EXCEL 2000. The DIV ID tag uniquely identifies each data table, so that the correct data table may be replaced when the user chooses to republish. Each item published will thus have a unique identifier generated at the time of publication. This unique DIV ID tag is incorporated into the HTML document and is also stored in the EXCEL 2000 spreadsheet from which the data are derived.

Preferably, the data table inserted into an HTML file will be surrounded by DIV tags. That is, information published to a file will be surrounded by <DIV> tags as shown here:

```
<DIV ID=identifier>
  Publishedinfo
</DIV>
```

where:

identifier is a unique string generated by EXCEL 2000, and

Publishedinfo is the data table transformed into HTML when published by EXCEL 2000.

Preferably, the DIV ID is a unique string generated by EXCEL 2000. In EXCEL 2000 the DIV ID string is "workbook.xls_random number," where workbook is the name of the workbook from which the data are being published, and the "random number" is a number up to five digits long that is randomly generated by EXCEL 2000.

If more than one data table is published to the same HTML document, EXCEL 2000 will be able to easily locate and republish the correct chart or data table by using the unique ID in the DIV ID and Style ID. As noted above, once generated, the unique DIV ID is incorporated into the HTML document, as well as stored in the parent EXCEL 2000 Workbook file from which the chart or data table was selected. Thus, when this EXCEL 2000 Workbook file is accessed and changes are made to the data table, these changes may be easily republished in the HTML document at the section of the HTML document corresponding to the EXCEL 2000 data table that has been changed.

Once the DIV ID has been generated at block 40, the logic proceeds to a block 42 in which the user specifies the destination file location. Note that the HTML document may be stored locally, or at a remote location. The user must provide input to a dialog box in EXCEL 2000 indicating the location of the document. This step leads to a decision block 24 in which the logic determines whether the indicated destination file exists. As mentioned earlier, it is possible that the HTML document into which the data table will be published may already be an existing HTML document, or it may be an HTML document which will be created by the act of publishing the EXCEL 2000 data table and converting into HTML compatible format. At decision block 24, if an HTML document does not already exist, the logic leads to a block 26 in which a new HTML document is created. Note that if in block 40 the user does not enter a destination file location, then in decision block 24, the logic will determine that the destination file does not exist.

The logic then proceeds to a block 28 in which a head section of the new HTML document is written out. Preferably, the head section will include comment information to make the newly written HTML more understandable to users viewing the HTML in an editing mode, such as:

<HEAD> tag: In cases where a new file is created, a <HEAD> tag is generated that includes a META tag for content type, language, generator, title, keywords, subject, etc.

<DIV > tag: The <DIV> tag is used so that republishing the Excel data is supported.

<small><h1 style="color: black; font-family: default font; font-size: 14 pt;

font-weight: 800; font-style: normal;"></small><small> Author's title is inserted here.</small><LINK> tags for related and supporting documents.

<TITLE> tag with filename.

For example, if the user publishes a range of data from Sheet 1 in Workbook 1.xls to create a new HTML file, the head section would appear as:

<HTML>

<HEAD>

<META> tags for content type, language, generator, title, keywords, subject . . .

<TITLE> text from Publish Wizard</TITLE>

</HEAD>

Once the head section is completed at block 28, the logic proceeds to a block 30 in which the body of the HTML document is written out, along with any comments. It is important to note that the functionality of the data is stored in this section of the HTML document. When translating the EXCEL 2000 data into HTML, all of the relevant formatting information is translated into a style tag included in the HTML document. This step is critical so that when the data are reintroduced into EXCEL 2000 from the HTML document in a "round trip," all of the formatting required by EXCEL 2000 for proper data manipulation will be included. It is also important to note that the body of the HTML document written by EXCEL 2000 may include information relative to using an ActiveX control to enable the viewer of the HTML document displayed in a browser to actually manipulate the data, not just view the data.

Preferably, the ActiveX control is only written into the HTML being published or republished if the user has actively selected that "interactive" HTML be exported. The term "interactive" is used to denote the spreadsheet functionality of a data table in an HTML document being viewed

in a browser obtained by using an ActiveX control. The term "static" HTML refers to HTML with no ActiveX control component. In the preferred embodiment, static HTML is the default selection unless the default is overridden by the user. Such a selection is preferably made in conjunction with publish/republish decision block 16, which was discussed above. At decision block 16, EXCEL 2000 will provide the user with an opportunity to select interactive verses static HTML translation of the data.

The use of an ActiveX control is well understood in the computer arts, and those of ordinary skill in the art will readily appreciate how the ActiveX control may be used in accord with the present invention to provide "interactive" data tables in HTML documents. Further information regarding the ActiveX control may be found in the following reference: David Chappell: Understanding ActiveX and OLE, Redmond, Wash: Microsoft Press, 1996. ISBN 1-572-31216-5.

Having completed writing out the body of the new HTML document in block 30, the logic flows to a data table writing subroutine 15. This data table writing subroutine is shown in greater detail in FIG. 3, and will be discussed below with reference to that Figure. Once the logic has completed data table writing subroutine 15, the logic proceeds to a block 34, in which the file is closed, causing a closing file subroutine 17 to be implemented. The details of the closing file routine are shown in FIG. 4, and again will be discussed in detail with reference to that Figure. Once closing file subroutine 17 is finished, the publish (or republish) process is complete.

If in decision block 24 the logic determines that a destination file exists, the logic moves to a decision block 44 in which the logic is required to determine if the destination file is local or stored at a remote HTTP/FTP site. If the destination file is local, the logic proceeds to a block 48 and the destination file is opened. If the destination file is not stored locally, the logic proceeds to a block 46, and the file is downloaded. Once the destination file is downloaded, the logic proceeds to block 48, which indicates that the destination file is opened. Thereafter, the logic proceeds to a parsing subroutine 11 in order to insert the data into the HTML document, which is described in detail in FIG. 2. Once parsing subroutine 11 is complete the logic flows to block 34, and the file is closed (using close file subroutine 17) thus completing the process.

Returning to decision block 16, the logic determines whether the user desires to publish or republish. If the user desires to republish, the logic flows to a block 20, which retrieves the selected HTML document. The logic then proceeds to a block 22, to retrieve the unique ID associated with the selected data table in EXCEL 2000, which is to be republished (as noted above, when a selection from Excel 2000 is published, a unique ID is generated and incorporated into the HTML document and also stored in the source EXCEL 2000 Workbook file). As discussed above, the ID is used by EXCEL 2000 to determine the location in the existing HTML document at which to insert the data being republished. EXCEL 2000 creates a unique ID by appending a 5-digit randomly generated number to the workbook name. The 5-digit number is guaranteed to be unique for all publications made from that source workbook. It should be noted that the ID is common to both the data in the source Excel 2000 Workbook file and in the corresponding data in the HTML document. The DIV and Style tags for the data will use this unique identifier in their ID attributes, except that the Style tag ID attribute will have "_Styles" appended to the identifier. The 5-digit portion of the identifier will also be used to create style class names for the data to be

published in order to minimize duplicating style class names that may already exist in the HTML document. Once the ID is determined, the logic proceeds to decision block 24. Both possible exits from decision block 24 have been fully described above.

EXCEL 2000 will insert a data table into an existing HTML document in one of three locations. A first possible location (for either publish or republish) is at a matching DIV ID/Style ID tag marker tag as discussed above. A second possible location is at a table marker tag (`<!--##Table##-->`). A third possible location is immediately preceding the close body tag (`</body>`) of the HTML document. Preferably, the location choice is selected based on the following rules:

If the file contains a DIV tag whose ID matches the DIV ID for the item being republished, replace data within the data table associated with that DIV tag.

If the file contains multiple DIV tags whose ID matches the DIV ID for the item being published, replace only data within data table associated with the first DIV tag encountered.

If the file doesn't contain a DIV tag whose ID matches the published item's DIV, but does contain the string `<!--##Table##-->`, replace that string with the published data.

If the file doesn't contain a DIV tag with a matching ID or a `<!--##Table##-->` tag, but it does contain a `</BODY>` tag, insert the published data prior to the `</BODY>` tag.

If the file doesn't contain a DIV tag with a matching ID, a `<!--##Table##-->` tag, or a `</BODY>` tag, alert the user that the data cannot be published to this document.

It is contemplated that locations within an HTML document for inserting the data table or other data may also be selected by the user. For example, instead of inserting the data table immediately preceding the close body tag (`</body>`) of the HTML document, the data table can be inserted immediately following the start body tag (`<body>`). Such a location is generally not preferred, as this location represents the beginning of the HTML document displayed to a viewer in a Web browser and is thus generally reserved for introductory text or images. Any location in an HTML document that is associated with a recognizable tag can be used as a predefined data insertion point. The close body tag (`</body>`) is preferred because it should exist in any HTML document, and the insertion of a data table at that location (the end of the HTML document) is less likely to disrupt the format or "look" of the HTML document. It should be noted that EXCEL 2000 does not support insertion of the data at these other locations, although this feature may be desirable.

FIG. 2 provides details of parsing file subroutine 11, which is used whenever the data selected from EXCEL 2000 is to be published or republished into an existing HTML document. EXCEL 2000 retrieves the HTML document, opens it, and then parses the HTML document in order to copy the existing HTML. The new HTML representing the data selected in EXCEL 2000 is inserted in the proper position within the copied HTML document, replacing any previously published EXCEL 2000 data (for which the ID of the new data matches the old data). This process employs a new temporary file. EXCEL 2000 will begin parsing and copying the existing HTML document to the temporary file until the parsing process determines that the location at which the new HTML is to be inserted has been found, at which point EXCEL 2000 stops copying the HTML document. The new HTML compatible data representing the

selected EXCEL 2000 data are written to the temporary file, and then copying of the HTML document resumes after the end of the previous data that is now replaced and continues until the end of the document is reached. If this step is completed successfully, the temporary file is renamed, effectively replacing the previous HTML document.

When the logic determines that parsing subroutine 11 is required, the process is started at a block 54 in FIG. 2, so that text being parsed is copied to the temporary new file. The logic then proceeds to a decision block 56 in which the logic determines whether the end of the HTML document has been reached. If the end of the HTML document has been reached, the logic proceeds to block 34 of FIG. 1, and the HTML document is closed. If in decision block 56 the logic determines that the end of the HTML document has not been reached, the logic proceeds to a decision block 60.

At decision block 60, the logic determines whether an open DIV tag (`<DIV>`) or style tag (`<ID>`) with a matching ID has been reached during the parsing of the document. If so, the logic proceeds to a block 84. At block 84, the logic stops copying the existing HTML document to the temporary new file and proceeds to a block 88, which provides for parsing beyond the open DIV tag (`<DIV>`) or style tag (`<ID>`) with matching ID found in decision block 60, until the associated close tag (`</DIV>`) is found. The logic then proceeds to data table writing subroutine 15, which is shown in FIG. 3. Thus, when a selected EXCEL 2000 data table is translated into HTML, it is inserted into the HTML document at the end of the close tag (`</DIV>`) associated with an already published EXCEL 2000 data table. It should be noted that the already published EXCEL 2000 data table will be completely replaced by the new data table created by data table writing subroutine 15, as all of the information relative to a data table published from EXCEL 2000 is completely contained between the `<DIV>` and `</DIV>` tags.

After data table writing subroutine 15 is complete, the logic proceeds to a block 78, and copying of the existing HTML document is started once again. In this manner, the newly written data converted to HTML compatible form and representing the selected data table from EXCEL 2000 is inserted at the correct location within the HTML document. From block 78, the logic proceeds to block 54, and parsing of the original HTML document continues until the end of the document is reached.

At decision block 60, the logic determined whether at that point in the HTML document an open DIV or style tag with matching ID had been found. The above section describes the response if such a matching ID is found. If no matching ID is found, the logic flows to a decision block 62, to determine whether the instant location in the HTML document is at a close body tag (`</body>`). If a close body tag has been reached, a decision block 66 determines whether any new data has already been written. If new data has been written, the logic flows back to block 54, and parsing of the HTML document continues. In decision block 56, the logic determines if the end of the HTML document has been reached.

If at decision block 66 the logic determines that no new data has been written, the logic flows to a decision block 68. At decision block 68, the logic determines whether or not a table marker tag (`<!--##Table##-->`) was found while the HTML document was being parsed. If the table marker has been seen, the logic moves to a block 70. At block 70, the logic re-initializes the file pointer and sets the flag to insert the newly written HTML representing the selected data table from EXCEL 2000 at the table marker tag found. The logic then returns to parsing and copying the HTML document in

11

block 54. If at decision block 68 the logic determines that no table marker tag has been found, the logic proceeds to a block 74.

As discussed with respect to FIG. 1, the newly written data in HTML compatible format representing the data table selected in EXCEL 2000 will be inserted in one of three possible locations. The decision blocks in FIG. 2 define the logic used to make that determination. For example, if at decision block 68, the logic determines that a table marker tag was seen in the HTML document, it can be concluded that although a table marker is present, no matching ID exists within the HTML document (otherwise a prior decision block would have called for the insertion at the matching ID, and a different path would have been followed). According to the preferential insertion rules noted above, the logic should insert the newly written data in HTML compatible format representing the data table selected in EXCEL 2000 at the table marker tag. This result is obtained when the logic proceeds to block 70 from decision block 68 as a result of an affirmative response to the decision inquiry. It should be recalled that one of the insertion rules provides that if no table marker tag, no matching ID, and no close body tag are present in the HTML document, an error message will be sent to notify the user of this circumstance. In such a case, the logic followed in FIG. 2 would proceed from finding no matching ID at decision block 60, to finding no close body tag at decision block 62, to finding no table marker at decision block 64, and would then return to parsing the HTML document at block 54, exiting parsing subroutine 11 at decision block 56. Thereafter, the logic would proceed to closing file subroutine 17, which as noted below, includes an error message block to notify the user that none of the three above-listed elements have been found in the HTML document.

Returning to block 74 in FIG. 2, the logic stops copying the HTML document to the new temporary file. The logic then proceeds to data table writing subroutine 15, which, as mentioned above, is shown in FIG. 3. Once the EXCEL 2000 selected data table has been written into HTML compatible form, the logic returns to block 78 and starts copying the HTML document, as described above. From that point, the logic returns to block 54 and continues to parse the HTML document and to copy the HTML document to the new temporary file (until the end of the HTML document, at which point parsing subroutine 11 is exited).

Now returning to decision block 62, if the logic determines that the instant location being parsed in the HTML document is not at a close body tag (</body>), the logic continues to decision block 64. Decision block 64 determines whether or not the instant location being parsed in the HTML document is at the beginning of a table marker tag (<!--##Table##-->). If not, the logic returns to parsing and copying the HTML document (file) in block 54. If the instant location is at the beginning of a table marker tag, the logic proceeds to a decision block 80. Decision block 80 determines whether to insert the data in HTML compatible format representing the data table selected from EXCEL 2000 at the table marker tag (using the insertion rules as previously discussed with respect to FIG. 1 and with respect to decision block 68 of FIG. 2). If decision block 80 determines that the HTML data should not be inserted at the table marker tag, the logic flows back to step 54 and continues to parse and copy the HTML document. If at decision block 80 the logic determines that the EXCEL 2000 data table is to be inserted at the table marker tag, the logic flows to block 74, and the logic stops copying and writes the new data (as was discussed above with respect to decision block 68).

12

It should be noted that block 74 and block 84 are equivalent, but lead to slightly different logic paths. These blocks both require that the copying of the HTML document into a temporary new file be halted. Immediately after block 84, the logic proceeds to block 88, which requires the logic to keep parsing the DIV or style tag (found in the HTML document, having an ID that matches the data table selected in EXCEL 2000; see the discussion supra of decision block 60) until a matching close tag (</DIV>) is found. This step is not required in the logic path that leads to block 74, because no DIV or style tag was found in decision block 60; thus, there is not an open DIV or style tag to parse until a matching close tag is found. Both block 88 and block 74 lead to data table writing subroutine 15, which is shown in FIG. 3, so that the selected EXCEL 2000 data table is translated into HTML compatible format. Once data table writing subroutine 15 is completed, the logic returns to block 78, and the copying of the HTML document is resumed. The stop copying step described in block 74 and block 84 (and associated block 88), data table writing subroutine 15, and start copying block 78 work together to ensure that the newly written data in HTML compatible format representing the data table selected from EXCEL 2000 are inserted into the HTML document at the correct point.

When the logic determines that writing a data table from the EXCEL2000 format into HTML is required, the logic initiates data table writing subroutine 15. The logic proceeds to a block 92 in which the data table writing subroutine is initialized, and then to a block 94 in which the table tag is opened (<TABLE>). The logic then advances to a block 96 in which it iterates the rows in the selected range. In a block 98, the table row tag (<TR>) is opened. The logic will then proceed to a block 100, which provides for iterating any columns within the specified range. From there, the logic proceeds to a block 102 in which the table cell tag (<TD>) is opened.

A decision block 104 next determines whether a formula is present in the current cell. If a formula is present, the logic moves to a block 106, and the formula is parsed. From there, the logic flows to a decision block 108 to determine whether any references in the formula are out of the specified range. If no references are out of the specified range, the logic flows to a block 126, and the references to row/column as they appear in the HTML document are adjusted. For example, if the range selected by the user does not include cell A1 (the "first" cell in a table, located in the upper left hand corner) the output is adjusted so that in the table as displayed in the HTML document, the top left cell in the published range will be displayed as A1. The following illustration should clarify this point. A data table is displayed in EXCEL 2000 as follows:

In Excel	A	B	C	D
1				
2		10		
3			15	

If the user selects the range B2:C3 (indicated by the dark lines) from the EXCEL 2000 spreadsheet for publication, the converted data will be shown as

13

<u>In HTML</u>	<u>A</u>	<u>B</u>
<u>1</u>	10	
<u>2</u>		15

The underlined elements are shown for illustrative purposes only, and would not be displayed in the spreadsheet or in the HTML document.

After adjusting the references in block 126, the logic proceeds a block 128 in which any formula in a cell being processed is written in HTML as a comment tag, so that if the resulting data table portion of the HTML document is "round tripped" back into EXCEL 2000, all the formatting necessary for full functionality of the table cells in EXCEL 2000 will be retained and made available. The logic next flows to a block 110, which provides for writing the result data (from executing the formula) in HTML to a table cell tag (<TD>). The logic then proceeds to a block 112, for generating a close table cell tag (</TD>). A decision block 114 determines whether more columns are present. If no more columns are present, the logic flows to a block 118, which generates a close row tag (</TR>). Next, a decision block 120 determines whether any more rows are present. If no more rows are present, the logic flows to a block 122, which generates a close table tag (</TABLE>). At this point, the data table writing subroutine is completed.

Returning to decision block 104, if the logic determines no formula is present in the current cell, the logic proceeds to block 110 and writes the result data (the contents of the cell) in HTML. The logic then proceeds to block 112 and generates the close cell tag as described above (</TD>). If at decision block 108 the logic determined that a reference from the formula is out of range, block 110 again writes the result data (i.e., the displayed contents of the cells), and then the logic again proceeds to block 112 to generate the close cell tag. It should be noted that block 110 may be entered from either decision block 104, decision block 108, or block 128.

If at decision block 114 the logic determines that more columns are present, the logic returns to block 100 and iterates any columns in the row that within the selected range. Similarly, if at decision block 120 the logic determines that more rows are present, the logic returns to block 96 and iterate any rows remaining within the selected range, and then repeats the above logical steps until the data table writing subroutine is complete.

FIG. 4 illustrates the logical steps involved in closing file subroutine 17. Once the logic has determined that closing file subroutine 17 should be started, it advances to a decision block 134, which determines if new data has been written into the HTML file. If no new data has been written, the logic proceeds to a block 136 and aborts the process. An error message is displayed to the user, and then a block 150 deletes any temporary files. At that point closing file subroutine 17 is completed. The user has the option of re-selecting publish or republish and initializing the process again. For example, if the user initially chose republish, but if a matching ID tag, a table marker tag, or a close body tag was not found in the HTML document, it would be appropriate to start over again and select the publish option to create a new corresponding data table in the HTML document.

If at decision block 134 the logic determines that new data has been written, a decision block 138 next determines whether the HTML file was resident on the computer used

14

by the user or was retrieved from an HTTP or FTP site. If the HTML file was resident on the computer being used by the user, a decision block 140 determines whether the destination file exists. If the destination file exists, a block 142 next deletes the old file. From that point the logic flows to a block 144 and renames the new file(s) to the name of the file(s) just deleted. It should be noted that there may be multiple files to support images for charts and data caches for pivot tables. At this point, closing file subroutine 17 is completed.

If decision block 140 determines that no destination file exists, the logic proceeds to block 144 and renames the new file. At that point, closing file subroutine 17 is completed. If at decision block 138 the logic determines that the original HTML file was retrieved from an HTTP or FTP site, the logic advances to a block 148 in which the newly written HTML file is uploaded to the destination where it will be stored, completing closing file subroutine 17.

Exemplary Operating Environment

FIG. 5 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. As discussed above, a preferred embodiment of the Publish/Republish feature is implemented as part of a spreadsheet program (EXCEL 2000) that is executed by a personal computer or workstation. The application program comprises a plurality of program modules that include routines, programs, objects, components, data structures, etc., which perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor based or programmable consumer electronics, network person computers, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

With reference to FIG. 5, an exemplary system for implementing the invention includes a general purpose computing device in the form of a conventional personal computer 220, including a processing unit 221, a system memory 222, and a system bus 223 that couples various system components including the system memory to processing unit 221. System bus 223 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes a read only memory (ROM) 224 and random access memory (RAM) 225. A basic input/output system (BIOS) 226, containing the basic routines that helps to transfer information between elements within personal computer 220, such as during start-up, is stored in ROM 224. Personal computer 220 further includes a hard disk drive 227 for reading from and writing to a hard disk, not shown, a magnetic disk drive 228 for reading from or writing to a removable magnetic disk 229, and an optical disk drive 230 for reading from or writing to a removable optical disk 231 such as a CD-ROM or other optical media. Hard disk drive 227, magnetic disk drive 228, and optical disk drive 230 are connected to system bus 223 by a hard disk drive interface 232, a magnetic disk drive interface 233, and an optical disk drive interface 234, respectively. The drives and their associated computer readable media provide nonvolatile storage of computer readable instructions, data

15

structures, program modules, and other data for personal computer 220. Although the exemplary environment described herein employs hard disk 227, removable magnetic disk 229, and removable optical disk 231, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, RAMs, ROMs, and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored on hard disk 227, magnetic disk 229, optical disk 231, ROM 224, or RAM 225, including an operating system 235, one or more application programs 236, other program modules 237, and program data 238. A user may enter commands and information into personal computer 220 through input devices such as a keyboard 240 and a pointing device 242. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to processing unit 221 through a serial port interface 246 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port, or a universal serial bus (USB). A monitor 247 or other type of display device is also connected to system bus 223 via an interface, such as a video adapter 248. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

Personal computer 220 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 249. Remote computer 249 may be another personal computer, a server, a router, a network PC, a peer device, or other common

16

network node, and typically includes many or all of the elements described above relative to personal computer 220, although only a memory storage device 250 has been illustrated in FIG. 5. The logical connections depicted in FIG. 5 include a local area network (LAN) 251 and a wide area network (WAN) 252. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

When used in a LAN networking environment, personal computer 220 is connected to local network 251 through a network interface or adapter 253. When used in a WAN networking environment, personal computer 220 typically includes a modem 254 or other means for establishing communications over WAN 252, such as the Internet. Modem 254, which may be internal or external, is connected to system bus 223 via serial port interface 246. In a networked environment, program modules depicted relative to personal computer 220, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Exemplary HTML

The following samples of HTML are provided: (1) an HTML compatible format representation of a sample data table selected from EXCEL 2000 and published as a new HTML document in accord with the present invention; (2) a sample of an existing HTML document, and (3) an HTML compatible format representation of a second data table selected from EXCEL 2000, different than that in (1), appended to the end of the existing HTML document of sample (2), in accord with the present invention.

Sample 1

```
<html xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns:x="urn:schemas-microsoft-com:office:excel"
xmlns="http://www.w3.org/TR/REC-html40"><head>
<meta http-equiv=Content-Type content="text/html; charset=windows-1252">
<meta name=ProgId content=Excel.Sheet>
<meta name=Generator content="Microsoft Excel 9">
<link rel=File-List href="._Publish_files/filelist.xml">
<style id="Book2_4606_Styles">
<!--table
{mso-displayed-decimal-separator:"\.";
mso-displayed-thousand-separator:"\,";}
.xl154606
{padding-top:1px;
padding-right:1px;
padding-left:1px;
mso-ignore:padding;
color:windowtext;
font-size:10.0pt;
font-weight:400;
font-style:normal;
text-decoration:none;
font-family:Arial;
mso-generic-font-family:auto;
mso-font-charset:0;
mso-number-format:General;
text-align:general;
vertical-align:bottom;
mso-background-source:auto;
mso-pattern:auto;
white-space:nowrap;}
.xl224606
{padding-top:1px;
padding-right:1px;
padding-left:1px;
```

-continued

```

mso-ignore:padding;
color:windowtext;
font-size:10.0pt;
font-weight:400;
font-style:normal;
text-decoration:none;
font-family:Arial;
mso-generic-font-family:auto;
mso-font-charset:0;
mso-number-format:0%;
text-align:general;
vertical-align:bottom;
mso-background-source:auto;
mso-pattern:auto;
white-space:nowrap;}

-->
</style>
</head><body>
<!--[if !excel]>&nbsp;&nbsp;&nbsp;<![endif]-->
<!--The following information was generated by Microsoft EXCEL 2000's
Publish as Web
Page wizard.-->
<!--If the same item is republished from EXCEL 2000, all information between
the DIV
tags will be replaced.-->
<!--START OF OUTPUT FROM EXCEL 2000 PUBLISH AS WEB PAGE
WIZARD-->
<!--START OF OUTPUT FROM EXCEL 2000 PUBLISH AS WEB PAGE
WIZARD-->
<div id="Book2_4606" align=center x:publishsource="Excel">
<table x:str border=0 cellpadding=0 cellspacing=0 width=256 style="border-
collapse:
collapse;table-layout:fixed;width:192pt">
<col width=64 span=4 style="width:48pt">
<tr height=17 style="height:12.75pt">
<td align=right style="width:48pt; height=17" class=xl154606" style="width:64
style="height:12.75pt; width:48pt">Alpha</td>
<td align=right style="width:48pt; height=17" class=xl154606" style="width:64
style="height:12.75pt; width:48pt">Beta</td>
<td align=right style="width:48pt; height=17" class=xl154606" style="width:64
style="height:12.75pt; width:48pt">Gamma</td>
<td align=right style="width:48pt; height=17" class=xl154606" style="width:64
style="height:12.75pt; width:48pt">Delta</td>
</tr>
<tr height=17 style="height:12.75pt">
<td align=right style="width:48pt; height=17" class=xl1224606" style="width:64
x:num="00.43">43%
</td>
<td align=right style="width:48pt; height=17" class=xl1224606" style="width:64
x:num="00.86">86%</td>
<td align=right style="width:48pt; height=17" class=xl1224606" style="width:64
x:num="00.77">77%</td>
<td align=right style="width:48pt; height=17" class=xl1224606" style="width:64
x:num="00.61">61%</td>
</tr>
<tr height=17 style="height:12.75pt">
<td align=right style="width:48pt; height=17" class=xl1224606" style="width:64
x:num="00.03">3%
</td>
<td align=right style="width:48pt; height=17" class=xl1224606" style="width:64
x:num="00.55">55%</td>
<td align=right style="width:48pt; height=17" class=xl1224606" style="width:64
x:num="00.16">16%</td>
<td align=right style="width:48pt; height=17" class=xl1224606" style="width:64
x:num="00.29">29%</td>
</tr>
<tr height=17 style="height:12.75pt">
<td align=right style="width:48pt; height=17" class=xl1224606" style="width:64
x:num="00.59">59%
</td>
<td align=right style="width:48pt; height=17" class=xl1224606" style="width:64
x:num="00.71">71%</td>
<td align=right style="width:48pt; height=17" class=xl1224606" style="width:64
x:num="00.08">8%</td>
<td align=right style="width:48pt; height=17" class=xl1224606" style="width:64
x:num="00.4">40%</td>
</tr>
<!--[if supportMisalignedColumns]>
<tr height=0 style="display:none">
<td align=right style="width:48pt; height=0" class=xl1224606" style="width:64
style="height:0pt; width:48pt"></td>
<td align=right style="width:48pt; height=0" class=xl1224606" style="width:64
style="height:0pt; width:48pt"></td>
<td align=right style="width:48pt; height=0" class=xl1224606" style="width:64
style="height:0pt; width:48pt"></td>
<td align=right style="width:48pt; height=0" class=xl1224606" style="width:64
style="height:0pt; width:48pt"></td>
</tr>
<!--[endif]>
</table></div><!--END OF OUTPUT FROM EXCEL PUBLISH AS WEB PAGE WIZARD-->
<!--END OF OUTPUT FROM EXCEL PUBLISH AS WEB PAGE WIZARD-->
</body></html>
Sample 2

```

-continued

```

<html>
<head>
<title>HTMLDocument</title>
</head>
<body>
<h4>Existing stuff on some html page.</h4>
<center><img src=images/MontSt-Michel.jpg></center>
</body>
</html>

```

Sample 3

```

<html xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns:x="urn:schemas-microsoft-com:office:excel"
xmlns="http://www.w3.org/TR/REC-html40"><head>
<title>HTMLDocument</title>
<link rel=File-List href="/ExistingHTMLAfter_files/filelist.xml">
<style id="Book1_18074_Styles">
<!--table

```

```

{mso-displayed-decimal-separator:"\.";
mso-displayed-thousand-separator:"\,";}

```

.xl1518074

```

{padding-top:1px;
padding-right:1px;
padding-left:1px;
mso-ignore:padding;
color:windowtext;
font-size:10.0pt;
font-weight:400;
font-style:normal;
text-decoration:none;
font-family:Arial;
mso-generic-font-family:auto;
mso-font-charset:0;
mso-number-format:General;
text-align:general;
vertical-align:bottom;
mso-background-source:auto;
mso-pattern:auto;
white-space:nowrap;}

```

.xl1218074

```

{padding-top:1px;
padding-right:1px;
padding-left:1px;
mso-ignore:padding;
color:windowtext;
font-size:10.0pt;
font-weight:700;
font-style:normal;
text-decoration:none;
font-family:Arial, sans-serif;
mso-font-charset:0;
mso-number-format:General;
text-align:center;
vertical-align:bottom;
border:.5pt solid windowtext;
background:#99CCFF;
mso-pattern:auto none;
white-space:nowrap;}

```

.xl12318074

```

{padding-top:1px;
padding-right:1px;
padding-left:1px;
mso-ignore:padding;
color:windowtext;
font-size:10.0pt;
font-weight:400;
font-style:normal;
text-decoration:none;
font-family:Arial;
mso-generic-font-family:auto;
mso-font-charset:0;
mso-number-format:General;
text-align:general;
vertical-align:bottom;
border:.5pt solid windowtext;
background:silver;
mso-pattern:auto none;
white-space:nowrap;}

```

-->

```
</style>
</head>
<body>
<h4>Existing stuff on some html page.<h4>
<center><img src=images/MontSt-Michel.jpg></center>
<!--[if !excel]>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~>
<!--The following information was generated by Microsoft EXCEL 2000's
Publish as Web
Page Wizard.-->
<!--If the same item is republished from EXCEL 2000, all information between
the DIV
tags will be replaced.-->
<----->
<!--START OF OUTPUT FROM EXCEL 2000 PUBLISH AS WEB PAGE
WIZARD-->
<----->
<div id="Book1__18074" align=center x:publishsource="Excel">
<table x:str border=0 cellpadding=0 cellspacing=0 width=128 style='border-
collapse: collapse;table-layout:fixed;width:96pt'>
<col width=64 span=2 style='width:48pt'>
<tr height=17 style='height:12.75pt'>
<td colspan=2 height=17 class=xl2318074 width=128 style='height:12.75pt;
width:96pt'>From Excel</td>
</tr>
<tr height=17 style='height:12.75pt'>
<td height=17 class=xl2318074 style='height:12.75pt;border-top:none'>new
data</td>
<td class=xl2318074 style='border-top:none;border-left:none'>new data</td>
</tr>
<tr height=17 style='height:12.75pt'>
<td height=17 class=xl2318074 style='height:12.75pt;border-top:none'>new
data</td>
<td class=xl2318074 style='border-top:none;border-left:none'>new data</td>
</tr>
<tr height=17 style='height:12.75pt'>
<td height=17 class=xl2318074 style='height:12.75pt;border-top:none'>new
data</td>
<td class=xl2318074 style='border-top:none;border-left:none'>new data</td>
</tr>
<tr height=17 style='height:12.75pt'>
<td height=17 class=xl2318074 style='height:12.75pt;border-top:none'>new
data</td>
<td class=xl2318074 style='border-top:none;border-left:none'>new data</td>
</tr>
<tr>
<td colspan=2>
<!--[if supportMisalignedColumns]~>
<tr height=0 style='display:none'>
<td width=64 style='width:48pt'></td>
<td width=64 style='width:48pt'></td>
</tr>
<![endif]~>
</table></div><!------->
<!--END OF OUTPUT FROM EXCEL PUBLISH AS WEB PAGE WIZARD-->
<!------->
</body>
</html>
```

5/15/05, EAST Version: 2.0.1.4

23

- (f) parsing the HTML document until the unique ID tag for all of the translated original data that were published is found; and
- (g) replacing all of the translated original data with all of the translated modified data.
- 2. The method of claim 1, wherein the application program comprises a spreadsheet program and the data that are translated include one of a data table and a chart.
- 3. The method of claim 1, wherein the translated original data are inserted into an existing HTML document immediately preceding the close body tag (</body>) of the HTML document.
- 4. The method of claim 1, wherein the translated original data are inserted into an existing HTML document at a marker tag previously inserted into the HTML document.
- 5. The method of claim 1, wherein the HTML document does not yet exist, and is created by the act of publishing the translated original data as a new HTML document.
- 6. The method of claim 1, wherein the step of preserving the format of the translated original data comprises the step of storing formatting information for the translated original data within a style section of the HTML document.
- 7. The method of claim 1, wherein the step of inserting the translated original data selection into the HTML document comprises the step of using the application program to insert tags into the HTML document to indicate a beginning and end of the translated original data inserted into the HTML document, and to indicate functional elements of said data.
- 8. The method of claim 1, wherein the step of preserving the format of the translated modified data comprises the step of storing formatting information for the translated modified data within a style section of the HTML document.
- 9. A method for enabling data from a spreadsheet to be published into an HTML document, and subsequently republished into the HTML document to include changes made in the data within the spreadsheet, comprising the steps of:
 - (a) specifying an original data selection that is to be published, within the spreadsheet;
 - (b) translating the original data selection into a translated data having an HTML compatible format, said translated data including a unique identifier (ID)) tag so that the translated data may be readily located within the HTML document;
 - (c) incorporating the translated data into the HTML document;
 - (d) modifying the original data selection within the spreadsheet, producing modified data that are to be republished into the HTML document;
 - (e) translating the modified data to produce translated republished data having an HTML compatible format, said translated republished data including the unique ID tag;
 - (f) parsing the HTML document until the unique ID tag for the translated data is found; and
 - (g) replacing the translated data with the translated republished data.
- 10. The method of claim 9, further comprising the step of restoring either the translated data or the translated republished data back into the spreadsheet, wherein the steps of translating both the original data selection and modified data each further comprises the step of including information in the HTML document that enables the translated data and translated republished data, respectively, to retain their functionality when restored back into the spreadsheet.
- 11. The method of claim 10, wherein the step of restoring comprises the step of retranslating the translated data or the

24

- translated republished data in the HTML document back into a format required by the spreadsheet.
- 12. The method of claim 9, wherein the original data selection and the modified data comprise a chart.
- 13. The method of claim 9, wherein the HTML document into which the translated data are incorporated is an existing HTML document, so that the translated data are appended to an end of the existing HTML document.
- 14. The method of claim 9, wherein the unique ID tag comprises a randomly generated number.
- 15. The method of claim 14, wherein the unique ID tag further comprises a parameter associated with the original data selection or modified data, which is associated with the spreadsheet.
- 16. The method of claim 15, wherein the parameter is a workbook name.
- 17. A method for enabling data from a spreadsheet to be published into an HTML document, such that the data published into the HTML document are restorable back into the spreadsheet without loss of formatting and functionality within the spreadsheet, comprising the steps of:
 - (a) specifying a data selection within the spreadsheet that is to be published into the HTML document;
 - (b) translating the data selection into translated data having an HTML compatible format, but including information required for maintaining formatting and functionality of the data selection when the translated data are restored in the spreadsheet, said information being retained within a comment section of the HTML document; and
 - (c) automatically incorporating the translated data into the HTML document.
- 18. The method of claim 17, further comprising the step of creating the HTML document into which the translated data are published when the step of publishing occurs.
- 19. The method of claim 17, wherein the translated data are inserted into an existing HTML document.
- 20. The method of claim 19, wherein the translated data are appended to an end of the existing HTML document.
- 21. The method of claim 19, wherein the translated data are identified by a unique ID tag that is maintained with the translated data and stored with the spreadsheet, said unique ID tag indicating a location within the HTML document at which the translated data are inserted.
- 22. The method of claim 21, further comprising the steps of:
 - (a) enabling a user to change the data selection within the spreadsheet after it has been published into the HTML document;
 - (b) translating the data selection as changed into translated republish data having an HTML compatible format; and
 - (c) using the unique ID tag to locate the translated data within the HTML document, replacing the translated data in the HTML document with the translated republish data.
- 23. An article of manufacture adapted for use with a computer, comprising:
 - (a) a memory media; and
 - (b) a plurality of machine instructions stored on the memory media, said plurality of machine instructions, when executed by a computer, implementing a plurality of functions, including:
 - (i) enabling a user to specify an original data selection to be published as original translated data into an HTML document;

25

- (ii) translating the data selection into the original translated data having an HTML compatible format, said original translated data including a unique identifier (ID) tag so that the original translated data may be readily located within the HTML document;
- (iii) incorporating the original translated data into the HTML document in association with the unique ID tag;
- (iv) translating a modification of the original data selection into translated republished data having an HTML compatible format, said translated republished data including the unique ID tag;
- (v) parsing the HTML document until the unique ID tag for the original translated data is found; and
- (vi) replacing the original translated data with the translated republished data.

24. The article of manufacture of claim 23, wherein the application program is a spreadsheet program and the data selection comprises one of a data table and a chart.

25. The article of manufacture of claim 23, wherein the original translated data are inserted into the HTML document at a predefined location.

26. A system for enabling an application program having no HTML editing capability to directly publish a data selection into an HTML document, such that the data published into the HTML document are restorable back into the application program without loss of formatting and functionality within the application program, comprising:

- (a) a memory in which a plurality of machine instructions defining the application program are stored;

26

- (b) a display; and
- (c) a processor that is coupled to the memory to access the machine instructions and to the display, said processor executing said machine instructions to implement a plurality of functions, including:
 - (i) enabling a user to specify the data selection to be published into the HTML document;
 - (ii) translating the data selection into translated data having an HTML compatible format, said translated data including information required for maintaining formatting and functionality of the data selection if the translated data are restored into the application program, said information being retained within a comment section of the HTML document; and
 - (iii) inserting the translated data into the HTML document so that the translated data are viewable on the display.

27. The system of claim 26, wherein the application program comprises a spreadsheet program and the data selection comprises one of a data table and a chart.

28. The system of claim 26, wherein the machine instructions further cause the processor to enable changes in the data selection to be republished into the HTML document, by translating the data selection that has been changed into translated republish data having an HTML compatible format, and replacing the translated data with the translated republish data in the HTML document.

* * * * *